ALGORITHMS FOR MACRO AND MICRO PLANNING OF RETAIL SPACE IN MOTION PICTURE INDUSTRY

By

MUKESH BASANDANI

DEPARTMENT OF INDUSTRIAL AND MANAGEMENT ENGINEERING
Indian Institute of Technology Kanpur

FEBRUARY, 2002

ALGORITHMS FOR MACRO AND MICRO PLANNING OF RETAIL SPACE IN MOTION PICTURE INDUSTRY

A Thesis Submitted in

Partial Fulfillment of the Requirements

for the degree of

Master of Technology

by

Mukesh Basandani

to the

DEPARTMENT OF INDUSTRIAL AND MANAGEMENT ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

February 2002

CERTIFICATE

It is certified that the work contained in this thesis entitled "***Algorithms for Macro and Micro Planning of Retail Space in Motion Picture Industry***" has been carried out by **Mr. Mukesh Basandani (Roll No. Y011410)** under my supervision and this work has not been submitted elsewhere for a degree.

[illegible]

February, 2002

(Sanjeev Swami)

Assistant Professor
Industrial and Management Engineering
Indian Institute of Technology
Kanpur-208016
India

POST GRADUATE OFFICE
Submitted on 19/2/02
I. I. T. Kanpur - 16

ACKNOWLEDGEMENT

I wish to express my deep sense of gratitude and indebtedness towards Dr. Sanjeev Swami for his inspiring guidance, invaluable suggestions and constructive criticism. He was always a constant source of encouragement throughout my thesis work, and was more than a thesis supervisor.

I would also like to cherish the nice company of my friends Girindra, Himanshu, Shailendra, Kalyan, Vipul without which the life in lab would have been monotonous. I would like to extend my thanks to all the friends in hall 4 for making my stay at IITK, a smooth and memorable one.

I extend my thanks to IME faculty members and staff for showing a special bondage towards me as an IME family member.

- Mukesh Basandani.

CONTENTS

LIST OF FIGURES AND TABLES

ABSTRACT

Managing the allocation of shelf space for new products is a problem of significant importance for retailers. The problem is particularly complex for exhibitors—the retailers in the motion picture supply chain—because they face dynamic challenges, given the short life cycles of movies, the changing demand level over time, the scarcity of shelf space, and complex sharing revenue term between the exhibitor and distributors.

In the proposed research we address exhibitor's display space allocation problem in two stages. In Stage 1, named as macro scheduling, the exhibitor has to choose a subset of movies from consideration set (for an entire week) that would be run on multiple screens every week. We extend Moholkar's (2001) approach by taking into considerations real life constraints such as prior commitment and obligation period, and apply the proposed model to real world data. We also investigate the improvement of the proposed model over the existing approach being used by SilverScreener (Eliashberg, Swami, Weinberg and Wierenga, 2001).

In Stage 2, named as micro-scheduling, we propose the model to assist the theatre manger to decide the slots and screen on which movie should run during a particular day of the week. Part of the input to this stage is provided by Stage 1 planning in the form of list of movies to be considered. As the movie's demand also varies within a day depending on the genre of the movie, the generated movie schedule should be such that it can help the exhibitor to earn the maximum revenue while attempting to achieve various targets in customer satisfaction, balanced staff schedule and floor capacity utilization. An Integer Linear Programming algorithm to solve micro-scheduling problem is proposed. We show the application of the algorithm on a simulated data set and provide interpretation of various results.

CHAPTER 1

INTRODUCTION

Retailing, one of the important functions of marketing, is becoming an increasingly important area of management attention and academic research, especially in the developed countries. One of the major areas of research in retailing is the development of decision support models to help retailers improve their decision-making in the dynamic and complex environment. Most previous marketing research in this regard has been concerned with consumer durable goods, not with services or intangibles. This thesis seeks to advance our understanding of how quantitative model building can improve marketing decision making in complex dynamic environments by focusing on a perishable entertainment product, namely, movies. Though we focus on movies in this thesis, our methodology and results can readily be generalized to other entertainment products (e.g., performing arts, books, video games), travel services, fashion goods, and educational programs.

A movie is an interesting product for several reasons. Most movies are separate entities and can be considered an innovation because of the new features (e.g., actors, storyline, music, etc.) usually included in them. These new movies are released every week throughout the year. The product life cycle of movies is relatively short and is measured in weeks. The product is seasonal in nature and the prominent seasons are during holidays. Another interesting attribute of movies from a research standpoint, and a major component of the complexity associated with the problems addressed in this thesis, is their perishability, which is time based deterioration of demand/appeal of a movie. Thus a theater owner, with an objective of effective screen management faces a complex scenario. The complexity comes from various sources. First, the relatively large number of movies available ("Too many pix, too few screens," trumpets a *Variety* 1995 headline) combined with a short and decaying audience appeal over time poses a complex management challenge. The decision is further complicated because it is made for a number of screens in a multiple screen theater (i.e., a multiplex). Second, each week's release of new movies brings continual pressure from the distributors to generate screens and playtime for them. Third, exhibitors often possess a number of facilities (i.e., theaters) in the same geographical area. This presents another booking challenge – managing the interdependency among several facilities. Fourth, the nature of the distributor-exhibitor contract in the motion picture industry in the U.S. as well as in Europe is quite unique. For example, in the U.S.A., in signing a contract to play a movie in its theaters, the exhibitor becomes obligated to play the film for a certain period of time even when audience demand is weak. The financial arrangements between distributors and exhibitors are also unique to the motion picture industry. Box-office receipts are split between the distributors and exhibitors such that the split favors the distributor in the first few weeks of the movie playing, but shifts to the exhibitor's favor later on. Distributors thus have a strong incentive to promote the movie intensively in their initial play period. On the other hand, the longer exhibitor plays the movie, the larger its share of the box-office receipts becomes. At the same time, theater attendance for a movie typically declines the longer it plays. Generally, all concession revenues go to the exhibitor.

The complexity of the screen management problem just described indicates that there is a real need for a marketing management support system that can help theater programming managers in their task of optimally choosing movies for their limited screen capacity. These marketing management support systems (MMSS) have a high potential for helping managers, but an unpredictable chance to succeed. While many of its managerial problems tend to be fairly structured, the decision environment is quite dynamic, contractual arrangements between parties are complex, management turnover appears to be high, and perhaps most importantly the cognitive style of the decision makers is often non-analytical or heuristic (Wierenga, Van Bruggen, and Staelin, 1999) based. These characteristics represent challenges in developing implementable models for decision makers in this industry. Nevertheless, successful implementation of MMSSs in other areas of the arts and entertainment industry (e.g., Weinberg 1986) provides optimism for the movie industry.

In response to the dynamics and challenges posed by the complex characteristics of movies, a stream of research, particularly addressing the marketing of movies has contributed to the marketing literature. At the consumer behavior level, some of the research has questioned the relevance of the traditional information-seeking framework for studying the consumption of movies (e.g., Hirschman and Holbrook, 1982; Holbrook and Hirschman 1982). Another stream of research has focused on forecasting the enjoyment of movies at the individual level (Eliashberg and Sawhney 1994) as well as forecasting the commercial success of movies at the aggregate level (Smith and Smith 1986; Austin and Gordon 1987; Dodds and Holbrook 1988; Sawhney and Eliashberg 1996; Eliashberg and Shugan 1997). Additionally, some research has begun to emerge addressing diffusion (Mahajan, Muller, and Kerin 1984; Jones and Ritz 1991), seasonality (Radas and Shugan 1995), release timing (Krider and Weinberg 1998), clustering (Jedidi, Krider, and Weinberg 1998), sequential products (Lehmann and Weinberg 1998; Prasad, Mahajan, and Bronnenberg 1998), contract design (Swami, Lee, and Weinberg 1998), scheduling (Swami, Eliashberg and Weinberg 1999; Eliashberg, Swami, Weinberg and Wierenga 2001) and the impact of advertising (Zufryden 1996). This thesis advances the above stream of research in marketing of movies, by considering the problem related to retailing of movies.

Typically, perishability is thought of in terms of physical deterioration of a product such as a grocery item with an expiry date. In some sense, this view comes from the supply side of the product. In contrast, in this thesis we adopt a "demand side view" in the context of perishability. Thus, the physical product in the problems considered remains the same, but its demand perishes over time. In recent years considerable work has been done on the treatment of perishability in inventory control (e.g., Abad 1996, Jain and Silver 1994). In inventory control, perishability refers to the physical deterioration of units of a product. For the movies, we analogously define perishability as the decrease in the value/appeal of a movie with the passage of time. As mentioned earlier, previous research in shelf space management and dynamic decision making has mainly focused on non-perishable goods. In the context of movies, the complexity introduced by perishability in dynamic shelf space management problem can be summarized as which motion pictures to choose to show each week and for how long to play them. In case of exhibitors with multiple screens, issues such as allocation of different movies to different capacity screens, switching of the movies between screens and multiple screening of same movie add to the complexity of the general problem. This decision-making problem is further complicated by additional factors specific to the movies context. We discuss such factors in detail in the later chapters to follow.

The exhibitors, who are the retailers in movie market, also face the problem of limited "shelf-space" which is the available screens for scheduling various movies. In the proposed research we address exhibitor's problem in two stages. In Stage 1, the exhibitor has to select the movies which are to be run on multiple screens in the coming weeks. We assume an exponentially declining demand pattern of each movie for this multiple-screen problem. The complexity in such problems is because of multiple unequal capacity screens and the decay of movies which causes the exponentially declining demand pattern. With an assumption of equal screen capacities, the problem is similar to the conventional parallel machine scheduling problems. Some work has been done in this regard particularly focusing on the movie exhibitors (Swami, Eliashberg, and Weinberg 1999). However this assumption of equal screen capacities poses serious limitation in implementation of such models. Considering different capacities for screens introduces nonlinearity into the resulting model and makes it less tractable (Saxena, 2000). Further the application of Genetic Algorithms methodology (Holland 1975, Michalewicz 1992, Grefenstette 1986) based heuristic (Moholkar, 2001) showed improvement over other heuristics. We extend Moholkar (2001)'s approach by taking in to considerations other constraint like commitment and obligation period and applying the proposed model to real world data. We also investigate the improvement of the proposed model over the existing approach being used by SilverScreener.

In Stage 2 named as micro scheduling, we propose the model to assist the theatre manager to decide the slots and screen on which movie should run during a day. Part of the input to this stage is provided by stage1 planning in the form of list of movies to be consider. As the movie's demand also varies with in a day depending on the genre of the movie, the generated movie schedule should be such that it can help the exhibitor to earn the maximum revenue while satisfying the various constraints like customer satisfaction, balanced staff schedule and floor capacity constraint.

The rest of the thesis is organized as follows. Chapter 2 reviews the relevant literature. Chapter 3 explains how the retailing decisions are taken in motion picture industries and briefly explain the problems faced by the exhibitor at two stages. Chapter 4 discusses the first stage problem namely, macro scheduling, and various models and approaches to solve the problem. SilverScreener model proposed by Swami, Eliashberg, and Weinberg (1999) addresses the problem of exhibitors with multiple screens with assumptions of deterministic demand pattern of movies and equal capacity of all screens. Saxena (2000) explains the implementation of this model and proposes a conceptual nonlinear model to address this problem. Saxena (2000) also proposes a heuristic based on demand of individual movies, to address the problem of unequal screen capacities. Moholkar (2001) explains the GA based heuristic and compares its performance against Saxena's (2000) Screen Allotment heuristic. We propose an extension of Moholkar's (2001) approach to incorporate some real world constraints such as prior commitments and obligation periods for movies. We apply proposed approach to real world data and validate the results by comparing it with the SilverScreener's results and further investigate the improvement of the proposed model over SilverScreener implementation approach (Eliashberg, Swami, Weinberg and Wierenga, 2001) for randomly generated scenarios. ANOVA is performed to study the impact of various factors on the improvement achieved by proposed model over SilverScreener approach.

In Chapter 5, we discuss the second stage problem, namely micro scheduling problem. In this chapter, we discussed the various issues, which are being faced by the exhibitors of multiplex theatre in scheduling the movies on different screens of the theatre *during a given day.* Several conceptual models have been proposed to address the problem. The testing of the finally selected model has been done on the simulated data and interpretations of results have been provided.

In Chapter 6, we conclude by discussing the limitations and directions for future research.

CHAPTER 2

LITERATURE REVIEW

Previous researchers have recognized the interface between marketing and operations as an important research domain in the quantitative modeling research stream. In the operations management literature, Acquilano and Chase (1991, p. 17) mention that marketing specialists need an understanding of what the factory can do relative to meeting customer due dates, product customization, and new product innovation. In service industries, marketing and production often take place simultaneously, so a natural mutuality of interest should arise between marketing and OM [operations management]. Karmarkar (1996) stresses the need to do more integrative research between marketing and operations management. This thesis addresses such needs and uses both marketing and operations management tools in solving management problems.

2.1 Integer Programming Approach

A linear programming problem in which some or all the variables must take non-negative integer values is referred to as integer linear programming problem. When all the variables are constrained to be integer, it is called a pure integer-programming problem, and in case only some of the variables are restricted to have integer values, the problem is said to be a mixed integer-programming problem. In some situations each variable can take on the values of either zero or one; such problems are referred to as zero-one programming problems.

Integer programming is a valuable tool in operations research having a good potential for applications. Such problems occur quite frequently in business and industry. All the assignment and transportation problems are integer-programming problems. In these types of problems the decision variables are either zero or one.

In all such situations, the decision variable X_j,

$X_j =$ 1 if jth activity is performed,

0 if jth activity is not performed

In addition, all allocation problems involving the allocation of men and machine give rise to integer programming problems, since such commodities can be assigned in integers and not in fractions.

The integer programming problems are solved by either the Cutting Plane or Branch and Bound method. In Cutting Plane method by Gomory (1958), we first solve the integer programming problem as ordinary L.P. problem and then introduce additional constraints one after the other to eliminate certain parts of the solution space until an integral solution is obtained.

In Branch and Bound method, the problem is first solved as a continuous L.P. problem ignoring the integrality condition. If in the optimal solution some variable, say X_j is not an integer, then

$$X_j^* < X_j < X_j^* + 1$$

where X_j^* and $X_j^* + 1$ are consecutive non-negative integers. It follows that any feasible integer value of X_j must satisfy one of the two conditions, namely

$$X_j < X_j^* \text{ or } X_j > X_j^* + 1.$$

These two conditions are mutually exclusive and when applied separately to the continuous L.P. problems, form two different sub-problems. Thus the original problem is "Branched" into two sub-problems. Each of these sub-problems is then solved separately as a linear program, using the same objective function of the original problem. If any sub-problem yields an optimal integer solution, it is not further branched. However if it yields a non-integer solution, it is further branched into two sub-problems. This branching process is continued until each problem terminates with either integer valued optimal solution or there is evidence that it can not yield a better one.

2.2 Parallel Machine Scheduling

A number of machines in parallel are a setting that is important from both the theoretical and practical points of view. From the theoretical viewpoint, it is a generalization of the single machine. From the practical point of view, it is important because the occurrence of the resources in parallel is common in the real world. One may consider scheduling parallel machines as a two step process. First, one has to determine which jobs are to be allocated to which machines; second one has to determine the sequence of the jobs allocated to each machine (Baker 1993, Pinedo 1995).

In the parallel machine scheduling, the objectives are specified in terms of two performance measures, which are as follows.

2.2.1 Minimizing Makespan

The makespan C_{max}, defined as max $(C_1, .., C_n)$ where $C_1,\ldots C_n$ are the completion time of the jobs, is equivalent to the completion time of the last job to leave the system. A minimum makespan usually implies a high utilization of the machine(s).

McNaughton (1959) presented an elementary result for the makespan problem when the jobs are independent and preemption is permitted. With preemption allowed, the processing of a job may be interrupted and the remaining processing can be completed subsequently, perhaps on a different machine.

The minimum makespan M* is given by

$$M^* = \text{Max}\ \{\frac{1}{m}\sum_{j=1}^{n} t_j,\ \max_j [t_j]\}$$

Where m = number of machines

n = number of jobs

t_j = processing time of job j

2.2.2 Minimizing Mean Flow Time

The other performance measure is minimizing the mean flow time. The flow time is the amount of time job j spends in the system. Flow time is given by,

$F_j = C_j - r_j$ and mean flow time is given by

$$\overline{F} = \frac{1}{n}\sum_{j=1}^{n} F_j$$

where C_j is the completion time and r_j is the release time.

Flow time measures the response of the system to individual demands for service and represents the interval a job waits between its arrival and its departure. In the current study, our objective is neither of the two objectives discussed above, but to maximize the total cumulative revenues generated over the time horizon.

2.3 Shelf Space Management Problems

Shelf (display) space management is a critical issue to the retailers. First, consumers choose from the products that are displayed on the shelf. In this sense, shelf space management determines the ultimate profitability of the retailer. Second, most retailers have limited shelf space to display their products. Therefore, the choice of which products to display becomes an important retailing decision. The decision is particularly complex in those industries in which many new products are continually introduced in market. Finally, anticipating and adapting to dynamic changes in consumer's tastes and demand are key concerns to most retailers. Moreover, such decisions have to be made for several product categories. Though previous researchers (Corstjens and Doyle 1983; Bultez and Naert 1988; Borin, Farris, and Freeland 1994) have addressed some of these issues, they have generally been in the context of packaged goods in a supermarket chain setting. However, other industry settings, such as the motion picture (or movie) industry considered in this thesis, pose different challenges and intriguing problems.

2.4 Perishability

One of the interesting and important attributes of the motion pictures is their perishability. In recent years considerable work has been done on the treatment of perishability in inventory control (e.g., Abad 1996, Jain and Silver 1994). In inventory control, perishability refers to the physical deterioration of units of a product. For the movies, we analogously define perishability as the decrease in the value/appeal of a movie with the passage of time. As mentioned earlier, previous research in shelf space management and dynamic decision making has mainly focused on non-perishable goods. In the context of movies, the complexity introduced by perishability in dynamic shelf space management problem can be summarized as: *which motion pictures to choose to show each week and for how long to play them.* This decision-making problem is further complicated by some other factors specific to the movie context. We discuss such factors in detail in the later chapters.

2.5 Decision Support Models in Marketing

Over the last three decades, decision support modeling has flourished in marketing. Today one can name many successfully implemented decision support models in marketing, such as PROMOTIONSCAN (Abraham and Lodish 1993), Rangaswamy, Sinha and Zoltners's (1990) model on sales force restructuring, SHARP (Bultez and Naert 1988), ARTS PLAN (Weinberg and Shachmut 1972), BRANDAID (Little 1972), CALLPLAN (Lodish 1971), and so on[1]. These models have addressed various aspects in marketing at different levels of a supply chain. For example, CALLPLAN efficiently allocates salesforce to customers and products. ARTS PLAN helps manager plan a series of performing arts presentations. PROMOTIONSCAN helps manager in developing and evaluating short-term retail promotions. SHARP model helps retailers decide on shelf-space allocations. The SilverScreener model (Swami, Eliashberg, and Weinberg 1999), presented in Chapter 4, is similar to PROMOTIONSCAN and SHARP models, its aim is to help retailers (of the motion picture industry) improve their decision-making.

Marketing decision support systems (MDSS) (Little 1979, p. 9) are intended to assist decision-makers in taking advantage of available information. Decision-makers should benefit from the availability of more or better data by incorporating the information derived from these data into their decision processes (Blattberg and Hoch, 1990). For this purpose MDSS contain marketing models that make it possible to perform so-called what-if analysis (Wierenga et al. 1994). Blattberg and Hoch (1990) report that a combination of model and manager often outperforms either of these two alone. Hoch (1994) attributes this to the relative strengths of models, which compensates for the relative weaknesses of managers. In general, some of the ways in which the use of decision support models can aid a marketing executive are the following (Montgomery and Weinberg 1973):

- Helps to better utilize a manager's judgment,
- Requires an explicit listing of input assumptions which leads to more informed discussion,
- Provides a method for quick and convenient evaluation of the consequences of alternative plans,
- Allows the emergence of unexpected solutions which open up new areas of problem-solving,

[1] Many commercial models based on the similar concepts are now also available in the market.

- Expands the range of questions which can be answered by use of the notion of derived judgment,
- Distills from available data relevant information as in new product forecasting,
- Provides a basis for relating marketing inputs to market results and, hence, serves as basis for marketing planning, and
- Diagnoses, based on early data, the adequacy of a market plan and locates areas needing improvement.

Inspite of these benefits, one must recognize the following important points about decision support models. First, models are an aid to the decision-maker, not a replacement. Marketing models often can help the manager make a better decision, but models do not make executive decisions by themselves. Second, models should be proposed as useful tools to the end-user (i.e., manager), not as academic curiosities. Finally, models can be useful to managers in many different ways, and may offer opportunities for efficiency in a broad range of managerial activities.

2.6 Genetic Algorithms

The Genetic Algorithm approach was first proposed by Holland (1975). The Genetic Algorithm seeks to mimic the approach of nature in the evolution of species, that is, it is based on the principles of population genetics to guide the search which results in "survival of the fittest'. This approach requires the specification of the candidate solutions in a binary or a nonbinary string format. These strings are analogous to chromosomes in nature. A chromosome in turn is composed of genes, each of which can take a number of values called alleles. Thus, for instance, we have the hair color gene at a specific position or locus in the string, which can take a specific allele value black, brown or red. In each generation (iteration) the selection of candidate solutions to participate in the creation of offspring (new candidates) is based on their ability to survive in the competitive environment (that is, its fitness). The GA approach has defined operators to perform activities corresponding to the creation of new chromosomes as a result of mating and chromosome modification as a result of mutation.

GAs have been applied in variety of fields. Goldberg (1989) has used GA for optimization of pipeline systems; Cleveland and Smith (1989) for scheduling flow shop releases; Liepins and Potter (1991) use GA for multiple fault diagnosis; Balakrishnan and Jacob (1996) use GA for product design; Deb, Chakraborty and De (1999) use GA for model based object recognition; Joshi (2000) uses GA in search space of hypercubes.

CHAPTER 3

RETAILING DECISION IN MOTION PICTURE EXHIBITION INDUSTRY

In retailing retail selling space is a fixed resource. Managing this space means making frequent decision about which product to stock and how much shelf space to allocate those products. A lot of research has focused on the strategic aspect of retail management-both with in the distribution channel and between retailers (or retailer manufacture systems)-but another important area has been the development of decision support system to help retailers improve their decision making. For example, Bultez and Naert's (1998) SHARP model helps retailers decide on shelf space allocations and Abraham and Lodish's (1993) PROMOTIONSCAN system helps managers in developing and evaluating short term retail promotions.

Most published research in this area focuses on consumer packaged goods sold through supermarkets. However, other retail formats and industry settings pose different challenges and intriguing problems. Our research focus on the products which have relatively short life cycles, so that effective retail management requires regular attention to the issue which product is to stocked. We focus our research on the motion picture industry.

Hollywood's major studios turnout is more than 450 feature films a year. Although the U.S. and Canada have 27,000 theater screens, many films go out of rotation before they reach their entire audience. Many films do not recoup their production costs even after accounting for overseas markets, cable and television sales or video rentals. During peak months -- May through September -- major studios planned 59 films in 1997, up from 49 the previous year.

Today the studios recognize the scarcity of shelf space but individual companies are finding it harder to find a screen for the films they want. Exhibitors have to make a master plan three or four months before a movie is even available. At that time, they can not predict how long it will run. The industry is just too dynamic and complex.

Hence the motion picture industry is emerging as an area of increased interest to marketing scholars and researchers. A stream of research, addressing various aspects related to marketing of movies, has begun to emerge in the marketing literature. At the consumer behavior level, some of the research has questioned the relevance of the traditional information seeking framework for studying the consumption of movies (Hirschman and Holbrook 1982). Another stream has focused on forecasting the enjoyment of movie at the individual level (Smith and Smith 1986, Austin and Gordon 1987, Dodds and Holbrook 1988, Sawhney and Eliashberg 1996, Eliashberg and Shugan 1997). Additionally, some research has begun to emerge addressing diffusion (Mahajan et al. 1984, Jones and Ritz 1991), release timing(Krider and Weinberg 1998), clustering (Jedidi et al. 1998), and the impact of advertising (Zufryden 1996), all in the context of motion pictures.

A theatre manager with an objective of screen management faces a complex scenario. The complexity comes from various sources. First, the increased supply of movies by various studios increases the difficulty of deciding which movie to play. This decision is further complicated by because it is made for a number of screens theatres. Second an additional supplie's of movies brings more pressure from the studios to guarantee sufficient play time for their movies. Relationship management in the motion picture industry is considered by many as very crucial. On the other hand, the scarcity of "shelf space" requires special attention in managing the screens effectively and profitably. Third, the nature of the distributor exhibitor contract in the motion picture industry is unique. In signing a contract to play a film in its theatre, the film for a certain period of time, even when the consumer demand is weak. This minimum obligation period, which is negotiated between two parties, may vary by movies as well as by studio. The financial arrangement between exhibitors and studios are also apparently unique to the motion picture industry. Unlike wholesale and retail pricing practices commonly employed in the consumer goods industry, box offices grosses are split between the exhibitors and distributors of motion pictures. The manner in which the box-office grosses are split favors the distributors in the first few weeks of the movie playing and, but shifts to the exhibitor's favor later on. Distributors thus have a strong incentive to promote the movies in their initial play period. On the other hand, longer the exhibitor plays the movie, the larger becomes his share of the box-office receipts. At the same time, theatre attendance for a movie typically declines the longer it plays.

In the face of this complexity, the current thesis provides the structure for analyzing the problems of exhibitors in two stages. As shown in Figure 3.1, in Stage 1, the SilverScreener approach (Eliashberg, Swami, Weinberg and Wierenga 2001) is used for generating the macro schedule. This schedule contains the list of movies, which are to be run in the coming weeks on different screens of any given theatre. The schedule is sent to Operation Planning Manager of theatre central planning office. He/she forwards this schedule to different theatre manager, who decides on the basis of that schedule which movie will run in which time slot during a day.

In general, macro scheduling operates as follows. Before a season begins, exhibitors select a set of movies to book and develop a tentative schedule (or "master plan"). Approximately three to four month before the summer season, distributors screen their movies for exhibitors at a major trade show. After the screening distributors send out bid application to most exhibitors. A typical bid invitation contains the release date of the movie, the contract terms (i.e. obligation period and sharing terms) that vary by both movies and distributors, bid return deadline, and so on. The cover letter, which accompanies the bid invitation letter, might feature a brief synopsis of the movie, along with the name of stars, stars director, producer and writer. Using the information about the contract term for various movies from the distributors' bid invitation letters, and a suitable revenue prediction scheme, the exhibitor can come up with a tentative preseason schedule for a multiplex using the model. Such a schedule can help the exhibitor decide whether or not bid for a particular movie. This preseason schedule would reject some movies outright, either because their contract terms are too unattractive, or their estimated profit potential is not attractive enough as compared to other movies.

When the season starts an adaptive scheduling approach, that is repeated weekly application of SilverScreener algorithm helps the exhibitor to decide how long should movie play. In practice, exhibitor decide once a week (normally on Monday), on which movies to play starting later in the week (usually on Friday). This is an adaptive decision making mode of behavior. In such situations the exhibitor is likely to choose a shorter planning horizon than the one for preseason bidding because he will be more certain of

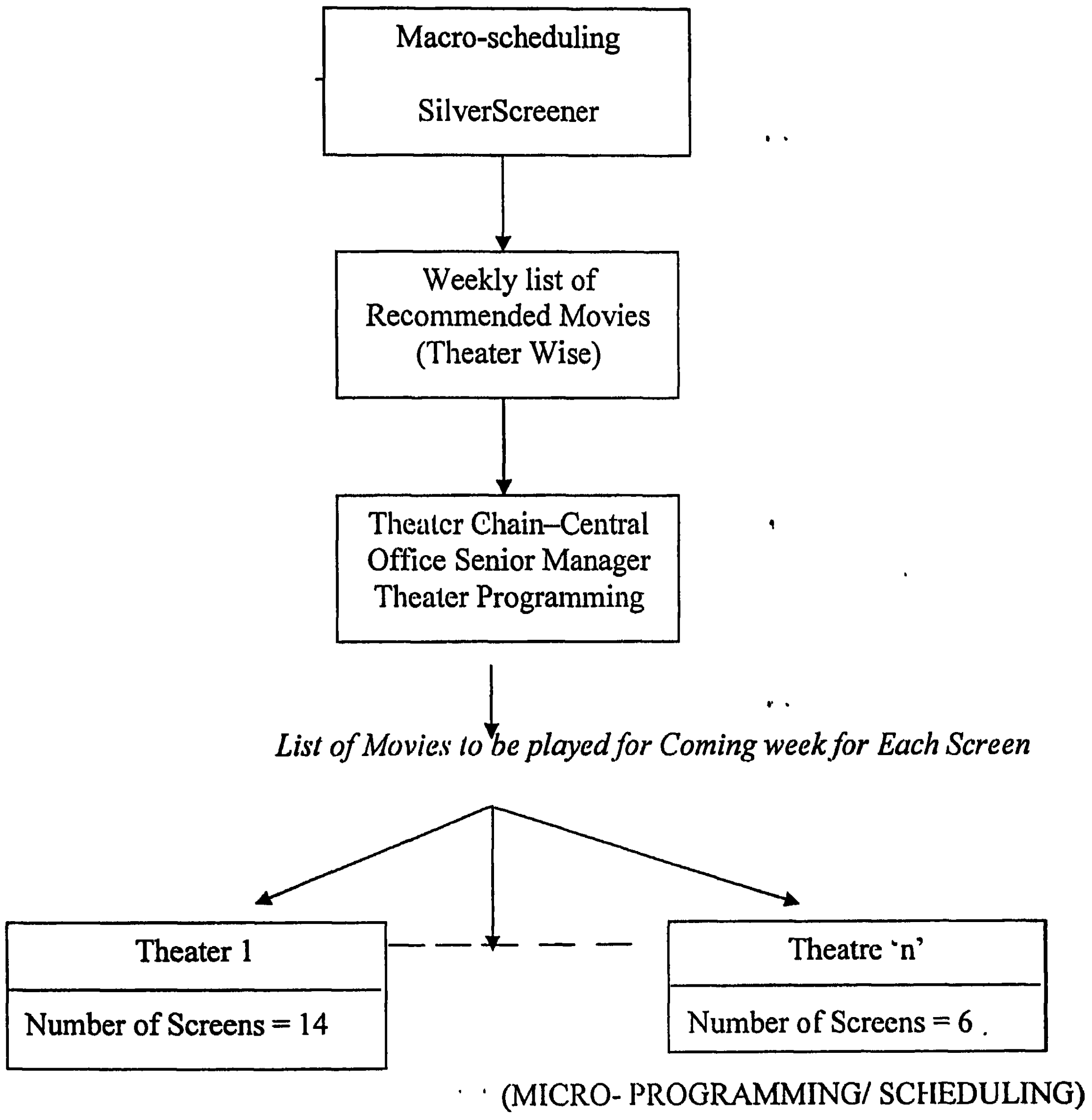

Macro-scheduling
SilverScreener
Weekly list of Recommended Movies (Theater Wise)
Theater Chain–Central Office Senior Manager Theater Programming
List of Movies to be played for Coming week for Each Screen
Theater 1
Number of Screens = 14
Theatre 'n'
Number of Screens = 6
(MICRO- PROGRAMMING/ SCHEDULING)

Fig 3.1: Macro and Micro-Planning of Movies in a Multiplex Theater

the availability and revenue potentials of the various movies. The exhibitor makes weekly decisions rolling from one time window to another. After a movies plays at the theater (if it is chosen to play), and the corresponding actual revenue data becomes available, the exhibitor can update the revenue prediction on the basis of the new data. In current thesis, macro scheduling refers to the adaptive scheduling mode of macro decision making.

Micro Scheduling: The above schedule gives the list of movies which should run in the coming week on different screens of the multiplex theatre. The movies which are recommended to run on the different screens of the theatre are from different genre; hence may have different demand pattern during a day. The demand is also different on week days and weekend days. The recommended schedule in the macro scheduling is on the basis of expected weekly demand. Hence this macro schedule needs to be adjusted according to the demand of movie within a day. This micro scheduling helps the exhibitor to plan that which movie he/she should play, in which time slot and on which screens, so that he/she can maximize his revenue and at the same time can address some other issues such as customer satisfaction.

CHAPTER 4

MACRO- SCHEDULING PROBLEM

4.1 Introduction

Most of the movie theatres in the major cities in the world are multiplexes, that is, they have more than one screening rooms. The exhibitor is interested not only in the optimal use of any one of his/her screens but has to optimize the use of all the screens. We propose to address these issues in this chapter. We present use of Genetic Algorithm based heuristic in solving the multiple-screen problem, for which simple mathematical programming approaches do not provide easily tractable exact solutions. Some interesting issues arise in case of a multiple-screen theater. Since the capacity for each screen is different, the decision of whether or not to schedule a movie, is accompanied by a decision of which movie should be allocated to which screen. If only demand of a movie is considered, the allotment may not be optimal because of some other factors like contract term between the exhibitor and the distributor. Again, with the varying demand for each movie and a varying percentage of total revenue that the exhibitor will retain, the movies may have to be switched between screens. The issues like multiple screening of movies (i.e., the same movie displayed on more than one screen in a given week) are other interesting issues arising in this case.

The chapter is organized as follows. In the next section we discuss the previous work specific to the problem. In Section 4.3, we explain in detail the problem formulation and briefly explain the SilverScreener model developed by Swami, Eliashberg, and Weinberg (1999) which deals with a similar problem. In Section 4.4, we explain the limitations of this model with the multiple-screen formulation proposed by Saxena (2000). We explain by an illustrative example, the situations in which the heuristic proposed by Saxena (2000) might give inferior solutions and justify the use of some other heuristic. In Section 4.5, we begin with an explanation of the Genetic Algorithm (GA) based heuristic proposed by Moholkar (2001). Further we propose an algorithm which extends Moholkar (2001)'s approach to more realistic problems by incorporating additional constraints, and then applies it to real world data. We attempt to validate the results of this approach to real data set. In Section 4.6, we studied its performance with capacity constraint over SilverScreener approach. In Section 4.7, we discuss the managerial implications and conclusions of our study. We also discuss the directions for future research.

4.2 Background of the Problem

Swami, Eliashberg and Weinberg (1999) deal with the problem of equal capacity multiple-screen theater scheduling. They propose an integer programming approach to this problem. Saxena (2000) extends their work by implementing the SilverScreener at a theater chain in The Netherlands. Saxena (2000) also addresses the limitation of model, which assumes equal capacity of all screens and proposes a conceptual non-linear model to relax the assumption. To address the issue of unequal screen capacities, Saxena (2000) proposes a screen allotment heuristic. Moholkar (2001) proposed a GA based heuristic for this problem and compares the results of his heuristic with Saxena's (2000) screen allotment heuristic.

4.3 Problem Formulation

4.3.1 The Exhibitor Problem

Every week, motion picture exhibitors have to make an important decision regarding the replacement of the movies playing at the screens in their theatres. The dynamic environment thus induced gives rise to the notions of decay and aging of movies. Decay is the intrinsic weekly decline in the box-office attraction and gross revenues (grosses in industry jargon) of a movie playing at a theater (Krider and Weinberg 1998). Aging is the decline in the value, that is, gross generating power, of a movie from an exhibitor's perspective if there is a delay (by week) in exhibiting the movie at the theater. Aging, therefore, results in an opportunity cost of not being able to play a particular movie.

The above scenario is further complicated by the nature of the contract between the distributor and the exhibitor. The basic structure of the contract is fairly standard between different distributor-exhibitor pairs although the individual terms may vary depending on the relationship between the two parties. A typical exhibition contract states a fixed obligation period and a differential revenue sharing scheme in different weeks between the distributor and the exhibitor. The obligation period limits the ability of an exhibitor to replace a movie with less than satisfactory box-office performance in the initial weeks after its release.

In a given week, the revenue sharing scheme splits the gross of a movie between a distributor and exhibitor by one of the two rules: a) 90%/10% over house nut[2] with minimum gross percentage, or b) a movie type based contract term where the percentage revenue share for distributors and exhibitors depend on the type of movie (Swami, Lee and Weinberg, 2000). If the *90%/10% over house nut* rule operates, then the distributor receives 90% of the gross after the exhibitor has deducted and retained the house nut amount. Accordingly, under this rule the exhibitor keeps 10% of the gross over house nut plus the house nut amount. The exhibition contract also contains minimum percentage figures as specified by the distributor for every week of the expected play length of a movie. These figures will be used if the *minimum gross percentage* sharing rule is invoked for revenue sharing. Under this rule, the whole gross box-office revenue amount (without house nut deduction) is split according to the specified percentage for that week. The splitting terms (in favor of distributor and exhibitor, respectively) specified by the distributor under a typical contract may appear as follows (see, for example, Squire (1992), p. 315, for the movie *Batman*).

90%/10% over approved house allowance with minimums of:

First week at	70%/30%
Next two weeks at	60%/40%
Next week at	50%/50%
Next week at	40%/60%
Balance at	35%/65%

The manner in which the box-office grosses are split favors the studio (distributors) in the first few weeks of the movie playing, but shifts to the exhibitor's favor later on. Distributors thus have a strong incentive to promote the movies intensively

[2] House nut is a small, negotiated amount, which the exhibitor receives from the distributor. It does not necessarily bear any relationship to the theater's actual expenses, and is only meant to allow for some cushion in the exhibitor's profit margins.

in their initial play period. On the other hand, the longer the exhibitor plays the movie, the larger becomes his/her share of the box-office receipts. At the same time, theater attendance for a movie typically declines the longer the movie plays.

In addition to the revenue earned from the box-office gross, the exhibitor also earns some income from concession sales such as popcorn, candies, and soft drinks. The concession sales, however, depend on individual demands generated by the movies playing at the theater. The exhibitor does not share the concession income with the distributor. Most theaters have multiple movies playing at their multiple screens. Given the complexity of the revenue sharing scheme and the dynamics of movie availability and decision making, the managers of such multiplexes are faced with non-trivial decisions of selecting and scheduling movies on different screens in a fixed planning horizon. In the next sub-section, we discuss a mathematical model SilverScreener proposed by Swami, Eliashberg, and Weinberg (1999).

4.3.2 SilverScreener: The Screens Management Model

4.3.2.1 Assumptions

Swami, Eliashberg, and Weinberg (1999), in order to simplify the exposition, make the following assumptions.

1. The availability of the movies to be released during the planning horizon is known in advance,
2. The weekly revenues to be generated by the movies considered during the planning horizon can be estimated in advance,
3. The replacement decisions are made on a weekly basis,
4. All the screens in the multiplex are of equal capacity,
5. There is no time lag between placing an order for a new movie and its arrival,

Assumptions 1, 3 and 5 are not limiting and, in fact reflect current industry practice. Assumption 2 is a strong assumption about *a priori* knowledge of movie revenues. However, data collected from *Variety* suggest that managers have a reasonable estimate of box-office gross revenue of a movie. Equal capacity screens are assumed (Assumption 4), which addresses the questions regarding which movies to pick and how long to show them. In the model implementation section, we discuss Saxena (2000) approach to this assumption.

4.3.2.2 Definition of Variables

W	- length of planning horizon,
H	- number of screens in the multiplex,
N	- total number of movies considered during the planning horizon,
r_j	- Release date of movie j,
d_j	- due date (if applicable) of movie j,
x_{jiw}	- binary (decision) 0-1 variable which takes value 1 if movie j is scheduled for i weeks beyond its obligation period starting in week w,
P_{jiw}	- profit received by the exhibitor if x_{jiw} is equal to 1,
$GROSS_{jw}$	- box-office gross revenue generated by movie j in week w,
POP_{jw}	- concession profit generated by movie j in week w,
VC_{jw}	- variable cost due to movie j in week w,
FC_w	- fixed cost of multiplex in week w,
$EXSHARE_{jw}$	- exhibitor's share of box-office revenue for movie j in week w,
OPD_j	- obligation period of movie j,
C	- house nut.

4.3.2.3 The Model

Time-Indexed Formulation

We now discuss *time-indexed formulation* (Sousa and Wolsey 1992; Williams 1997) that is particularly useful for solving the exhibitor problem. This formulation is based on the idea of dividing the planning horizon $[0, ..., W]$ into W discrete intervals of unit length. To model the screen management problem, a binary variable x_{jiw} is defined, which equals 1 if movie j is shown for i weeks *beyond its obligation period* starting in week w of the planning horizon, and 0 otherwise. Notice that the obligation period constraint is included in the definition of x_{jiw} itself. For example, if obligation period of Movie Number 3 is 2 weeks, then $x_{301} = 1$ implies that it is shown for 2 weeks starting in week 1.

The time-indexed formulation highlights some key differences between the exhibitor problem and typical machine scheduling problems. First, all movies do not have to be played in the exhibitor problem, while all jobs have to be scheduled in the machine scheduling problems. Accordingly, the proposed formulation is aimed at helping the exhibitor decide about two critical decision variables: *choice* of which movies and deciding *play lengths* of the chosen movies. Second, the following types of decision-making goals are found to be prevalent in machine scheduling problems: turnaround, timeliness, and throughput. In contrast, the exhibitor problem offers a situation in which the scheduling decisions directly affect profitability, a more relevant decision making goal, both by affecting gross revenue and concession profits. Finally, the main parameters of interest in machine scheduling are the lengths of the jobs, and their release and due dates. The exhibitor problem, on the other hand, deals with additional variables such as complicated contract terms and an exponentially decaying demand function.

Profit Function

P_{jiw}, the total profit the exhibitor receives corresponding to each x_{jiw}, is defined as:

$$P_{jiw} = \sum_{u=w}^{w+SCR_{ji}-1} (-FC_u + POP_{ju} - VC_{ju}) + I_{\theta_{ju}} * \{0.1 * (GROSS_{ju} - C) + C\} +$$

$$(1 - I_{\theta_{ju}}) * EXSHARE_{ju} * GROSS_{ju}, \tag{4.3.1}$$

$$j = 1, \cdots, N, \quad i = 0, \cdots, k_j, \quad w = r_j, \cdots, d_j - SCR_{ji} + 1.$$

where θ_{jw} is a logical condition given by

$$\theta_{jw} = (0.9 * (GROSS_{jw} - C) > (1 - EXSHARE_{jw}) * GROSS_{jw}),$$

and

$$\begin{aligned} I_X &= 1 \text{ if } X = TRUE \\ &= 0\text{, otherwise,} \end{aligned}$$

$k_j = d_j - r_j - OPD_j + 1$ = maximum possible number of weeks movie j can be shown *beyond its obligation period* starting in r_j or any *feasible* week thereafter,

$SCR_{ji} = OPD_j + i$ = total screening period for movie j if it is shown for i weeks *beyond its obligation period*, where $i = 0, \ldots, k_j$

The profit expression proposed by Swami, Eliashberg, and Weinberg (1999) consists of two portions corresponding to the two different conditions of the contract terms[3]. The two conditions, operationalized by the logical variable I, follow directly from the revenue sharing terms described earlier. In addition, there are variables for fixed and variable costs. Fixed costs (e.g., rent, lights, and weekly maintenance) are the costs that the exhibitor has to incur irrespective of the traffic generated by the movies playing in a particular week. Variable costs, such as salaries of the temporary staff hired for a particular movie, vary by movies as well as weeks.

The above operationalization of P_{jiw}'s simplifies the solution procedure considerably since they can now be computed indepen~~dently of the op~~timization routine. Further, the variables k_j and SCR_{ji} help us "cover" all feasible (P,x) pairs for a movie. For example, suppose movie j has parameters: $OPD_j = 2$, $r_j = 1$, and $d_j = 4$. If the movie is scheduled in Week 1 (i.e., $w = 1$), then it can be shown for 2, 3 or 4 weeks, that is, for 0, 1 or 2 weeks beyond the obligation period. Since $k_j = 2$, i varies from 0 to 2. The corresponding values of $SCR_{ji} (= 2 + i)$ are 2, 3, and 4, respectively. Therefore, the corresponding (P,x) pairs are (P_{j01}, x_{j01}), (P_{j11}, x_{j11}), and (P_{j21}, x_{j21}), respectively.

Demand Function

Swami, Eliashberg and Weinberg (1999), Krider and Weinberg (1998), and Sawhney and Eliashberg (1996) use an exponentially declining demand curve to estimate the number of visitors attracted by a movie.

Demand (number of visitors) = $VISITOR_{ju} = \alpha_j e^{\beta_j t + \varepsilon}$ (4.3.2)

where $\alpha_j > 0$ and $\beta_j < 0$ are opening and decay factors for Movie j,

[3] The SilverScreener does not consider the movie based contract term, however while comparing the performance of proposed GA-based heuristic with Saxena's (2000) heuristic, which uses SilverScreener in its first step, we made suitable modifications in the SilverScreener model to consider the movie based contract terms.

and $\varepsilon \sim$ normal $(0, \sigma^2)$.

Problem Statement

The time-indexed formulation of the exhibitor problem is presented as follows:

$$\max \sum_{j=1}^{N} \sum_{i=0}^{k_j} \sum_{w=r_j}^{d_j - SCR_{ji}+1} P_{jiw} x_{jiw} \qquad (4.3.3)$$

subject to

$$\sum_{i=0}^{k_j} \sum_{w=r_j}^{d_j - SCR_{ji}+1} x_{jiw} \le 1, \quad j = 1,\dots,N \qquad (4.3.4)$$

$$\sum_{j=1}^{N} \sum_{i=0}^{k_j} \sum_{q_j = w - SCR_{ji}+1}^{w} x_{jiq_j} \le H, \quad w = 1,\dots,W \qquad (4.3.5)$$

$$r_j \le q_j \le d_j - SCR_{ji} + 1, \quad j = 1,\dots,N; \; i = 0,\dots,k_j \qquad (4.3.6)$$

$$x_{jiw} \; \varepsilon \; \{0,1\} \qquad (4.3.7)$$

In the above model, Statement (4.3.3) denotes the objective function, which is to maximize cumulative profit over the season. Constraint (4.3.4) ensures that a movie is played in only consecutive weeks. It also allows a movie not to be scheduled at all. The next constraint restricts the total number of movies, scheduled in any week of the planning horizon, to the total number of screens in the multiplex. In doing so, it sums up all the movies, which are released earlier than or in the week under consideration. The set of inequalities denoted by Equation (4.3.6) is an indexing constraint, which restricts the variable q_j in Constraint (4.3.5) to feasible values. Constraint (4.3.7) defines x_{jiw} to be a binary variable.

4.3.2.3 SilverScreener Implementation at a European Theatre Chain

Screen Capacity

In their formulation Swami, Eliashberg and Weinberg (1999) assumed all the screens of equal capacity. However in the implementation of the model, Saxena (2000) followed the following simple heuristic for allocating the movies on different capacity screens.

Screen Allotment Heuristic (Saxena 2000)

The statement of the heuristic is:

In a particular week, allocate the movie with highest number of visitors to the highest capacity screen, the movie with the next highest number of visitors to the next highest capacity screen, and so on.

In other words, the application of the model first chooses a set of movies, and then the movies are allocated to the theater screens in the order of their capacities. The screen allocation heuristic is similar to the managerial decision making in some instances. Moholkar (2001) showed the limitation of this screen allotment heuristic by an example in which the movies having higher demand were assigned to the screens having higher capacity, which were giving lower revenue due to the complex contract term between exhibitor and distributor.

4.4 Multiple Screen Formulation (Saxena 2000)

SilverScreener model assumes all the screens of equal capacity (Assumption 4) for simplicity. The screen capacity is, of course, a critical variable to the problem formulation only if the demand of a movie exceeds screen capacity. Indeed, the primary reason for the exhibitor to have different screen sizes is that the attendance varies for different movies and having different sizes increases scheduling flexibility (while saving capital cost in the original construction of the theater). A movie may be switched from a higher to a lower capacity screen[4]. We now discuss the extension to the model as proposed by Saxena (2000) for an unequal capacity case.

4.4.1 Definition of Variables

T	- Length of planning horizon,
H	- Number of screens in the multiplex,
N	- Number of movies considered during the planning horizon,
r_j	- Release date of movie j
d_j	- Due date of movie j

[4] The opposite case may also occur when a movie's demand builds on word-of-mouth in the later weeks though rarely in practice.

OPD_j - Obligation period for movie j

cap_s - Capacity of screen s

x_{jsu} - Binary (decision) 0-1 variable which takes value 1 if movie j in scheduled on screen s in week u

y_{jw} - Binary (decision) 0-1 variable which takes value 1 if movie j in scheduled in week w

P_{jsu} - Profit received by the exhibitor if x_{jsu} is equal to 1

visitor[j,u] - number of visitors for movie j in week u

floor[j,u] - distributor's share in the box-office revenues for movie j in week u

4.4.2 Problem Formulation

Saxena (2000) defines P_{jsu}, the exhibitor receives corresponding to x_{jsu}, as

P_{jsu} = [Min {cap_s, *visitor* [*j, u*]}] * (2 + 13.5*0.89725*(1-*floor* [*j, u*]))

Since the implementation of SilverScreener by Saxena (2000) was in context of The Netherlands, the profit function is according to the practices of The Netherlands' motion picture industry. The above problem is formulated as an integer-programming problem. The time-indexed formulation is presented below.

Objective function: $$\text{Max} \sum_{j=1}^{N}\sum_{s=1}^{H}\sum_{u=1}^{T} P_{jsu}x_{jsu} \qquad (4.4.1)$$

Subject to:

$$\sum_{s=1}^{H} x_{jsu} \le 1 \qquad \forall \quad j, u \qquad (4.4.2)$$

$$\sum_{j=1}^{N}\sum_{s=1}^{H} x_{jsu} \le H \qquad \forall \quad u = 1..T \qquad (4.4.3)$$

$$x_{jsu} = 0 \qquad \forall \quad j, s \text{ and } d_j < u < r_j \qquad (4.4.4)$$

$$\sum_{s=1}^{H} x_{jsu+1} \le \text{Max}\left\{\left(1-\sum_{w=1}^{u} y_{jw}\right), \sum_{s=1}^{H} x_{jsu}\right\} \forall\ u=1..T\text{-}1, j \qquad (4.4.5)$$

$$\sum_{w=1}^{u} y_{jw} = \sum_{v=1}^{u}\sum_{s=1}^{H} x_{jsv} \qquad \forall \quad j, u = 1..T \qquad (4.4.6)$$

$$\sum_{j=1}^{N} x_{jsu} = 1 \qquad \forall\, s, u \qquad (4.4.7)$$

$$x_{jsu},\ y_{jw} \in (0, 1) \qquad (4.4.8)$$

In the above model, Constraint (4.4.2) ensures that in a particular week u any movie j can only be scheduled at most on one screen, that is, it cannot be scheduled on more then one screen in a given week. The inequality sign is to ensure that the movie may or may not be scheduled. For example if $x_{111} = 1$ i.e. movie 1 is scheduled on Screen 1 in Week 1 then the Constraint (4.4.2) $x_{111} + x_{121} + x_{131} + x_{141} + x_{151} + x_{161} \leq 1$ implies $x_{121} = x_{131} = x_{141} = x_{151} = x_{161} = 0$ as these are binary variables and can take the values 0 or 1.

Constraint (4.4.3) restricts the total number of movies scheduled in a given week to the total number of screens in the theater. Constraint (4.4.4) simply ensures that the movie can not be scheduled before its release date and after its due date. In most instances the movie runs in a continuous manner and it is very rare that the movie runs for few weeks and then resumes after a break of one or more then one weeks.

Constraint (4.4.5) deals with the continuity of the movie run. It restricts the movie to play in breaks. So this constraint schedules the movie in such a manner that the movie run is continuous. There are two situations that may arise. The examples for the situations are shown in detail in Moholkar (2001)'s thesis.

4.5 Genetic Algorithms Based Heuristic (Moholkar 2001)

Moholkar (2001) developed an approach for finding a schedule to maximize the total revenue the exhibitor will obtain, by the application of Genetic Algorithm. We provide here some background originally discussed in Moholkar (2001)'s thesis.

For the multiple-screen problem, GA views sequences of movies to be displayed on screens for each week, for the total planning horizon, as a chromosome (the candidate solution), which in turn is a member of a population. Fitness of each chromosome was measured by the value of the objective function (i.e. the total revenue earned by the exhibitor). The GA performance depends greatly on the choices of the different GA parameters, population size (p_s), the probability of crossover (p_c) and the probability of mutation (p_m). Design of experiments (DOE) approach was used to identify the good settings for p_s, p_c, and p_m before using GA to solve the problem.

An n x w vector to represents a chromosome for an n screen w week problem. For example, a chromosome for a two-screen two-week problem is represented as

m_1	m_5	m_1	m_3

The chromosome represents that Movie m_1 is played in Weeks 1 and 2 on Screen 1, Movie m_5 is played in Week 1 on Screen 2 and Movie m_3 is played in Week 2 on Screen 2. The GA begins with a population (population size p_s) of randomly generated vectors. The fitness of any chromosome is then evaluated based on the total revenue for that chromosome. To calculate the total revenue corresponding to a movie when placed in a particular week-screen slot, the number of visitors is taken as the minimum of the respective screen capacity and demand of the movie in that week. Using this number and the contract term, we calculate the revenue earned by that movie in that week. Total revenue is the cumulative of all such week-screen slot corresponding revenues. The revenue that the exhibitor earns from a week u screen s slot if movie j is scheduled in this slot is given by

$$revenue_{jsu} = [Min\{cap_s, demand_{ju}\}] * (1 - floor_{j,r}) \qquad \cdots(4.5.1)$$

Where $floor_{j,r}$ is percentage share of distributor for movie j, when run for weeks r and cap_s is capacity of screen s and $demand_{ju}$ is the demand for movie j in week u.

4.6 Enhancement of Genetic Algorithms Based Heuristic to Accommodate Real World Constraints

4.6.1 Motivation: Moholkar (2001) shows significant improvement in revenue the exhibitor can earn by focusing on capacity restrictions. However, this approach ignores real world constraints such as obligation period constraints and commitment constraints

Manager's commitments:

Under the term of contracts, exhibitor makes commitment with the distributor for certain movies that these will be played in specific screens of specific theatres. In the first phase of the study these commitments were not taken in to account. But the Management specifically wanted these to be honoured. Note specifically that in case of this constraint, it is interesting that partially the schedule is prefixed and the algorithm has to optimize over the rest of the schedule. As shown in Figure 4.1, the blocks filled by color are committed block and now the optimization is to be done for rest of the blocks.

	Screen			
		1	2	3
Weeks	1			
	2			
	3			
	4			

Figure 4.1: Optimization with Commitment Constraint

<u>Obligation period constraint</u>:

Under this constraint the exhibitor is obliged by the distributor to run the movie for at least its obligation period if it is selected. Hence the movies which have its run length less than its obligation period are either replaced by some other movie which has been scheduled in the previous week but not in the present week, or its run length is increased to obligation period by replacing the next week movie. Hence to decide which previous week movie should replace the current week movie or which next week movie should be replaced by the current week movie makes the situation interesting. As the Genetic Algorithm is search based heuristic, the chances of getting optimal schedule in this case is higher.

4.6.2 SilverScreener Approach to Address Additional Constraints

A scheme was devised to honor the commitments by SilverScreener. The minimum obligation period was taken one or two weeks for the week in which commitment was made. In later weeks of the planning window these commitment restrictions were relaxed and it was treated as normal movie, which would be recommended by the AMPL if the revenue stream was attractive enough to compete with the other movies in the consideration set.

For obligation SilverScreener fixes the obligation period one or two weeks in the input file for AMPL, in the output of the AMPL, movies which have been recommended for less than obligation period are either there run length is increased equal to obligation period or replaced by some other movie. Hence the changes are made manually in the output of the AMPL.

4.6.3 GA based Approach to Address Additional Constraints

4.6.3.1 The Model

The proposed model treats the two new constraints by adding them explicitly in the formulation. The definition of variable is same as used in Section 4.4

Objective function: $$\text{Max} \sum_{j=1}^{N}\sum_{s=1}^{H}\sum_{u=1}^{T} P_{jsu}x_{jsu} \qquad (4.6.1)$$

Subject to:

$$\sum_{s=1}^{H} x_{jsu} \leq 1 \qquad \forall \; j, u \qquad (4.6.2)$$

$$\sum_{j=1}^{N}\sum_{s=1}^{H} x_{jsu} \leq H \qquad \forall \quad u = 1..T \qquad (4.6.3)$$

$$x_{jsu} = 0 \qquad \forall \; j, s \text{ and } d_j < u < r_j \qquad (4.6.4)$$

$$\sum_{s=1}^{H} x_{jsu+1} \leq \text{Max}\left\{\left(1-\sum_{w=1}^{u} y_{jw}\right), \sum_{s=1}^{H} x_{jsu}\right\} \forall \; u=1..T-1, j \qquad (4.6.5)$$

$$\sum_{w=1}^{u} y_{jw} = \sum_{v=1}^{u}\sum_{s=1}^{H} x_{jsv} \qquad \forall \; j, u= 1..T \qquad (4.6.6)$$

$$\sum_{j=1}^{N} x_{jsu} = 1 \qquad \forall \; s, u \qquad (4.6.7)$$

$$\sum_{s=1}^{H}\sum_{u=1}^{T} x_{jsu} \geq OPD_j \qquad \forall \; j \qquad (4.6.8)$$

$$x_{jsu} = 1 \qquad \forall (j,s,u) \ni COMMIT \qquad (4.6.9)$$

$$x_{jsu}, y_{jw} \in (0, 1) \qquad (4.6.10)$$

In the above model the equation 4.6.8 ensures that for any given movie j the sum over planning horizon of all binary variable x_{jsu} will be at least equal to its obligation period. While the constraint 4.6.9 ensures that the movie which is in committed set, will be scheduled on the specific screen in specific week. Here the *COMMIT* is the three element set, in which first element corresponds to the committed movie, second element corresponds to the screen and the third to the week in which the movie is committed. For all the binary variable which have the subscripts matching with any one of the elements will take the value 1.

For the application of the GA we write its code in C language. This reads the input from a file, which is same as used by AMPL in SilverScreener with slight modification. It contains the revenue prediction for the movies in the different week of the planning horizon, release week of the movies and share data for the exhibitor for different movies, for different run length of the movies. The number of movies to be considered, number of screens in theatre, total number of weeks for which the planning is to be done are entered in the code as constant. In addition, the capacity of different screens and set of committed movies are entered manually in the program.

4.6.3.2 Initialization of the population:

GA begins with the randomly generated population of the solution (chromosome in genetic term). We also define $n \times w$ vector to represent the solution of the problem, as used by Moholkar (2001). The movie is selected randomly in the same manner except for the cells which were committed set are assigned a movie which are committed by management in that specific cell representing the screen and week. The week in which the committed movie is assigned is the release week of the movie. This randomly generated solution may not be feasible due to the complex constraint of continuity, constraint of obligation period. Hence each solution of the generation is repaired by the repairing functions for continuity and obligation period. The fitness of any chromosome is then evaluated based on the total revenue for that chromosome. To calculate the total revenue corresponding to a movie when placed in a particular week-screen slot, the number of visitors is taken as the minimum of the respective screen capacity and demand of the movie in that week. Using this number and the contract term, we calculate the revenue earned by that movie in that week. Total revenue is the cumulative of all such week-screen slot corresponding revenues. The fitness function used by us is same as the revenue function being used by SilverScreener in his approach. The revenue that the exhibitor earns from a week u screen s slot if movie j is scheduled in this slot is given by

$$Revenue_{ju} = 0.133 * [Min\{cap_s, demand_{ju}\}] + 0.9434 * [Min\{cap_s, demand_{ju}\}] * (1 - floor_{j,r} / 100) \quad (4.6.11)$$

Where $floor_{j,r}$ is percentage share of distributor for movie j, when run for weeks r and cap_s is capacity of screen s and $demand_{ju}$ is the demand for movie j in week u.

4.6.3.3 GA Parameters

This section will provide the brief overview of different genetic operator used in the current study building on Moholkar (2001). The application of the above methods in two parameters selection and crossover is also same except slight change in mutation. Hence here we will describe only the mutation operator and the two repairing functions for continuity constraint and obligation period constraint.

Mutation: The offspring produced after crossovers are subjected to mutation. Bit-wise mutation was used, a bit corresponding to each week-screen slot. Each bit can undergo mutation with a probability of mutation (p_m). A random number is generated corresponding to each bit, which is week-screen slot, if this random generated number is less than ($100*p_m$) and that week-screen slot is not committed by the manager than the movie in that slot is replaced by one of the movies, selected randomly, amongst those that are released till that week and yet not at all scheduled in the chromosome. If no such movie is available then one of the movies selected randomly, amongst those, which are not scheduled for that week but are released till then, is used for replacement. In another diversification which is done with a probability half that of mutation probability for each week-screen slot (if it is not committed bit), in which the movie at the current bit exchanges screen with a movie scheduled on some other screen for the same week. If the movie in some other bit which will exchange with the current bit is committed bit than next bit is selected, this is process is repeated till we get the bit which is not committed.

Repairing Function for continuity constraint: This function checks for a chromosome, whether the constraint of continuity of movie display is satisfied. For each discontinuously exhibited movie, the function can take one the three actions. First, to replace the "post-break" scheduling of the movie (should not be committed movie) by another movie which is scheduled in previous week but not in the current week. Second, to schedule the movie in the week, where it was not scheduled, causing discontinuity in exhibition, by replacing only such movie which was not scheduled in the previous week and is not in the committed week-screen slot but was scheduled in the current week. Third, to replace the "pre-break" scheduling of movie by a movie which was scheduled in the previous week but not in the current week. The repairing function takes the second action if the movie is scheduled in more than half number of total weeks, in the in-feasible chromosome. Otherwise if the numbers of exhibitions are more in "post-break" scheduling then repairing function takes third action, else it takes first action. Example is illustrated in Figure 4.2 (a).

Repairing Function for obligation period constraint: In order to get a feasible solution after each crossover and mutation this repairing function is used. This function checks for a chromosome, whether the constraint of obligation period is satisfied. For each movie which is appearing for the first time in each week-screen slot the run length is calculated. If this run length of the movie is less than the obligation period then this function can take one action out of the following two actions. First, to replace the movie in the next week by itself, if the next movie has the two characteristic first it is appearing for the first time or it has not been scheduled before that week in the solution, and it is not in the committed week-screen slot. Second we will replace the movie with the movie which has been appeared in the last week but not in the current week. The second action is taken if the repairing is being done in the last week of the planning horizon. The example is illustrated in Figure 4.2 (b).

Chromosome before Continuity Constraint Repairing Function (blocks with bold elements are committed blocks)

Screen

Weeks	1	2	3
1	m_7	m_5	m_{16}
2	$\mathbf{m_8}$	m_{10}	m_9
3	m_3	m_5	m_2
4	m_2	$\mathbf{m_6}$	m_5
5	m_2	m_5	m_{16}

Chromosome before Continuity Constraint Repairing Function (blocks with bold elements are committed blocks)

Screen

Weeks	1	2	3
1	m_7	m_5	m_{16}
2	$\mathbf{m_8}$	m_5	m_9
3	m_3	m_5	m_2
4	m_2	$\mathbf{m_6}$	m_5
5	m_2	m_5	m_6

Figure 4.2(a): Illustration of Repairing function for Continuity Constraint with commitment

Chromosome before Obligation Constraint Repairing Function (blocks with bold elements are committed blocks)

Screen

		1	2	3
Weeks	1	**m_7**	m_8	m_9
	2	m_8	m_{10}	M_9
	3	**m_{27}**	m_5	M_2
	4	m_2	m_6	M_5

Chromosome after Obligation Constraint Repairing Function

Screen

		1	2	3
Weeks	1	**m_7**	m_8	m_9
	2	m_8	m_{10}	M_9
	3	**m_{27}**	m_5	M_2
	4	m_2	m_{27}	M_5

Figure 4.2(b): Illustration of Repairing function for Obligation Period constraint

4.6.3.4 GA Parameter Optimization

The performance of GA based heuristics greatly depend upon the values of its parameters, namely, the population size p_s, the probability of crossover p_c, and the probability of mutation p_m. In our thesis work we also used the same *Design of Experiments* approach to optimize the three GA parameters. Each of these parameters has two settings for which the performance of GA is tested.

There are eight experiments that cover all combinations of settings for these three factors. The '0' setting for a factor indicates that it is set to a lower amongst the two values, and the '1' setting indicates that it is set to a higher value. For each experiment, the GA is run for a fixed number of generations and the revenue corresponding to the best schedule is noted down. Then for a factor, the sum of revenues corresponding to all the experiments in which the factor was set at '0' is compared with that of experiments where it was set to '1'. The setting of the factor with higher value for the sum is then considered better. This is done for all the factors. Now if setting '1' is proved to be better for a factor, we make it as a '0' setting and make some higher value as a '1' setting for the factor, thus moving in the direction of higher values. Similarly, if '0' setting proves better, we move in the lower direction for that particular factor. With these new settings the process is repeated. If moving in any direction from a setting for a factor degrades the performance of GA, we conclude the setting for the factor as optimal.

4.7 Performance Evaluation

SilverScreener Implementation (Swami, Eliashberg, Weinberg, Wierenga 2001) helps to select and schedule movies for a multiple-screens theater over a fixed planning horizon in such a way that the exhibitor's cumulative profit is maximized. The SilverScreener has successfully withstood the demands of the time. The theatre management's requests in the second phase of the multiple screen theater scheduling pilot study which starts on 6th August, 2001 were incorporated in the SilverScreener without requiring many changes. However this approach may lead to inferior solutions since first the movies to be scheduled are selected by the integer programming model, without considering the screen capacities and then the screens are allotted to the movies by considering only their demand. In allocating the screens to movies an important factor of contract term between the exhibitor and distributor is neglected.

The GA approach that we propose addresses such concerns. It is a search method, and if cleverly deployed, may lead to actual optimal solution. In this section we examine under which conditions of the input parameters, the proposed GA-based heuristic, which explicitly considers the capacity variations in the screens, may perform better than SilverScreener approach.

For evaluating the performance of the proposed heuristic with SilverScreener, we used the methodology similar to Moholkar (2001). Movies are classified into four "types" based on their opening strength and decay rate (Jedidi, Krider, and Weinberg 1998). The data of 84th St. Sixplex, a six-screen theater in New York, was used to estimate the opening strength and the decay rate of each of these types of movies. First, data were median split on the basis of the opening strength of movies. Then, each of these groups was median split on the basis of their decay rates. The median values of the data for each of the four groups were used as the representative opening strength and decay rates for the groups. We estimated the demand pattern for each type of movie by using the equation $\lambda * e^{\alpha - \beta t}$ where α represents the opening strength, β represents the decay rate, and t represents the number of weeks after the release of movie. The constant multiplier λ was chosen arbitrarily for appropriate scaling.

We first discuss the factors chosen for comparing the two approaches. The first factor considered for is the *number of committed movies* in a given scenario. This factor limits the scope of optimization of the problem; therefore its amount was varied to examine its impact. The second factor considered is *screen capacity* because it directly impacts the scheduling of a multiple screen theatre. We found in our preliminary analysis that SilverScreener has a tendency to select *Type I movies*[5] in their preliminary weeks because of their high opening strength. However with the added constraint of capacity, the value addition in the objective function corresponding to Type I movies might be reduced. Therefore, we select the *number of Type I* movies in a scenario as third factor to be analyzed.

[5] Type I movies are the movies with high opening strength but decay rapidly.

We examine two levels, high and low, of the capacity factor. To consider the typical distribution of capacity over screens, we use the screen capacity data of a theater in The Netherlands. In the high capacity case, we assume eight shows in each screen room per week and in low capacity we assume four shows per week[6]. Table 4.2 gives the capacities we assumed for each of the cases.

Table 4.1 Capacity of Screens

Capacity of Screen						
Case	1	2	3	4	5	6
High Capacity	3472	2736	1728	1208	1112	904
Low Capacity	1736	1368	864	604	556	452

For the second parameter, the number of committed movies in the scenario, we define two levels, high and low. High level scenario is one in which there are five or more than five committed movies and the scenario with no committed movies is defined as low level scenario. The numbers of committed movies are based on observing real world implementation data of SilverScreener.

To operationalize the proportion of Type I movies in a scenario, we classify the scenario as a *high 'decay property'* scenario in which there are ten or more rapidly decaying movies (Type I). Similarly, if there are three or less number of Type I movies, we classify the scenario as *low 'decay property'* scenario.

We used the fixed contract term for all movies as shown below.

Table 4.2 Movie based Contract term.

	Exhibitor's % Share in Total Revenue for Movie Type			
Weel	I	II	III	IV
1	10	15	25	25
2	30	20	40	40
3 onwards	50	35	50	50

The above data is according to the industry practice. Movies of Type I open with high demand and decay rapidly, so distributors of these movies have a tendency of retaining maximum share of revenue in their initial weeks. Since Type II movies retain their

[6] Again these numbers of shows do not exactly represent the actual number of shows in a theater, and are scaled according to demand of movies in initial weeks.

strength for long, the distributor offers less share of revenue from these movies to the exhibitors. Movies of Type III and IV open with low strength and never have less demand, so to promote exhibitors to play these movies, distributors offer more share in the revenue for such movies.

4.7.1 Simulation Setup

The input parameters required to generate problems for various simulation analyses are as follows:

i. length of planning horizon
ii. number of screens at the theatre
iii. quantity of movies (i.e., total number of movies considered) in a scenario
iv. quality of movies
v. release dates of movies

To generate different simulation problems, we assign fixed values to the first three parameters, namely, the length of planning horizon, the number of screens in a theater and the quantity of movies released during the season, which are common to all problems. We set the length of the planning horizon to be 8 weeks (e.g., representing the two peak months of summer season) and the number of screens to be 6. The quantity of movies released is also kept constant. We assume release of 4 movies in each week. To begin with we assume that there are six movies already playing on the six screens at the end of the previous season and that the exhibitor can continue to play any of them in the new season.[7] Therefore the movies released in week one of the season are not guaranteed of exhibition unless the exhibitor finds it more profitable to replace some of the currently playing movies with the newly released movies. The release dates of movies are thus fixed by assigning the number of movies released per week to 4.

To randomize the assignment of a movie to a movie type, we use the relative proportions of Type I (19%), II (7%), III (38%), and IV (36%) movies in Jedidi, Krider, and Weinberg's (1998) sample. Specifically, a random number is drawn among integers between 1 and 100 for each movie. Then the movie is assumed to be Type I if the

[7] Thus there are 38 movies in each simulated season.

random number is between 1 and 19, Type II if its between 20 and 26, Type III if its between 27 and 64, and Type IV if it is between 65 and 100.

We study the effect of capacity of the screens (high/low), number of committed movies, and decay property (high/low) for the period, on the percentage difference between exhibitor's cumulative revenue from proposed heuristic and SilverScreener approach. We use a complete 3 (treatment) X 2 (level) factorial design.

4.7.2 Methodology

Using the above scheme of simulating problems, a total of 70 problems were generated. Based on the input data, the problems were assigned to their respective type of experiment. The GA model of Section 4.5 and 4.6 was coded in C language (Appendix 2) and run on an Intel Pentium III class computer. The value evaluations were done according to the cumulative revenue criterion presented in Section 4.5.2

4.7.3 Simulation Results

Table 4.3 summarizes the simulation results in several ways. In the following table, factor A is represents the screen capacity factor, B represents the number of committed movies, the second factor and C represents the third factor decay property of the scenario. Table 4.3(a) gives the mean (and standard deviation) of improvement shown by GA-based heuristic over SilverScreener's approach, under each factor level combination. Table 4.3(b) gives the percentage improvement in each replication of the different treatment settings and mean improvement for each treatment, by GA-based heuristic over SilverScreener approach, for each main effect. Table 4.3(c) shows the average effect of each factor and the interactions among various factors. This table shows that the average effect of screen capacity is negative and high, which shows that the percentage improvement in GA based heuristic decreases as the screen capacity increases. In other words, both approaches will give almost the same result if screen capacity will increase to a very high level. Hence our approach is quite effective in cases in which screen capacity constraint is binding. Similarly for the second factor, the numbers of committed movies have a negative effect this means that as the number of committed movies increases the percentage improvement decreases. In other words, if there are too many commitments the scope for improvement by proposed heuristic over SilverScreener approach is limited. The ANOVA results in Table 4.3(d) show that no interactions are significant and that all three main effects screen capacity, number of committed movies and decay property are significant at the *.05* level.

Table 4.3(a): Mean Improvement in Cumulative Revenues by GA-based Heuristic over SilverScreener Implementation approach

Exp.	Capacity[a] (A)	Commi- -ted movies[b] (B)	Decay Property[c] (C)	Number of problems	Mean improvement in cumulative revenue*	% improv- ement
1	0	1	0	6	3244.5(556)	21.87
2	1	1	0	6	1975(715)	7.76
3	0	0	0	6	4159(338)	29.20
4	0	1	1	6	4007.77(205)	29.74
5	1	0	0	6	5098(608)	20.42
6	0	0	1	6	4763(632)	35.70
7	1	1	1	6	2979(941)	12.14
8	1	0	1	6	5020 (952)	21.57
				Total = 48**		

a – High (1) setting implies high capacity levels, otherwise low (0) setting
b – Setting high(1) implies that there are 5 committed movies, setting low (0) implies 0 committed movies
c – High (1) setting implies more than 9 type 1 movies in the scenario, Low (0) setting implies less than 4 type 1 movies.
* Standard deviation values are shown in parentheses.
** Out of 70 problems, only 48 problems could be assigned exactly to specific scenarios.

Table 4.3(b) Percentage improvement data on various setting of different parameter for calculating the effect of parameter

Parameters Settings			Replicates						Average Improvement
A	B	C	1	2	3	4	5	6	
0	1	0	28	23.75	25.18	17.48	18.8	18	21.87
1	1	0	7.26	8.5	9	12.9	3.7	5.22	7.76
0	0	0	24.74	26.59	32.63	28.45	32.49	30.3	29.20
0	1	1	29.18	32.13	25.54	30.48	31.13	30	29.74
1	0	0	20.8	16.65	20.85	23.73	20.5	20	20.42
0	0	1	24.92	37.31	46.52	31.68	35.63	38.14	35.70
1	1	1	9.5	16.52	9.2	12.63	18.57	6.4	12.14
1	0	1	15.3	22.84	30.11	18.36	20.62	22.2	21.57

. **Table 4.3(c) Average effect due to various parameters and interactions**

Parameters and their interactions	**Average improvement**
Capacity (A)	-13.65*
Committed Movies (B)	-8.84
Decay Property (C)	4.975
AB	-2.2*
AC	-2.21*
BC	1.15
ABC	0.4616

* Negative average effect shows that as the level of the factor increases the improvement of the proposed approach over SilverScreener approach decreases.

Table 4.3 (d): ANOVA Results for Improvement of GA-based Heuristic over SilverScreener Implementation Approach (2001)

Source of Variation	Sum of squares	Degrees of freedom	Mean square	F_0
Capacity (A)	2237.37	1	2237.37	120.397*
Committed Movies (B)	938.71	1	938.71	50.51*
Decay Property (C)	297	1	297	15.98*
AB	58.12	1	58.12	3.127
AC	58.76	1	58.76	3.16
BC	15.87	1	15.87	0.854
ABC	2.5576	1	2.5576	0.1376
Error	743.33	40	18.58	
Total	4351.72	47		

- Significant at 5 percent.

4.7.4 Discussion

Intuition suggests that the factor of capacity should be the most important factor in improvement of cumulative revenue by GA-based heuristic as compared to SilverScreener's approach. The reason is that the basic difference in the SilverScreener's approach and the proposed GA-based heuristic lies in treatment of capacity constraint. Our result shows that the screen capacity is the most significant factor. The negative average effect of this factor suggests that the utility of our approach is higher when the screens are of relatively lower capacities than the case when the capacity of screens is high. In other words if capacity constraints are tight, the theatre may lose revenues by assigning a high revenue movie to a screen for which the movies revenues exceeds the capacity.

The second factor found important is the number of committed movies. This also has the negative average effect, which shows that as the number of committed movies increases the improvement by the GA based approach will be lower than the case in which the numbers of committed movies are lower. This makes intuitive sense as the increased number of committed movies reduces the scope of optimization.

The third factor found important is the decay property of a season or the total number of rapidly decaying movies in a season. It is important to note that the setting for this factor were somewhat extreme settings. This may have introduced an artificial significance to this factor. Since Type I are the rapidly decaying movies, the distributor of such movies have the tendency of retaining maximum revenue, from the first few weeks exhibition of these movies. So the initial exhibitor share for such movies is less. Each movie of Type I scheduled by SilverScreener approach leads to degradation of the schedule, because first, the exhibitor share in profit is less and second, these movies are allotted the screens with maximum capacities because of their high demand. Therefore, the number of such movies in a season affects the improvement shown by GA-based heuristic over SilverScreener approach.

4.8 Application of the Proposed Approach to Real-World Data and Additional Benefits

The above results shows that under certain settings of parameters such as screen capacity, number of committed movies, number of Type I movies the proposed heuristic may perform better than the 2-stage SilverScreener approach used by Eliashberg, Swami, Weinberg and Wierenga, 2001. However the exact settings of these parameters may not be available in every application setting. In such cases, it would be interesting to examine whether the schedule generated by the proposed approach "matches" with the best implementation approach available so far. For this purpose, the proposed heuristic was applied to the data of a five-theatre chain in Europe being aided currently by SilverScreener (Eliashberg, Swami, Weinberg and Wierenga, 2001). In the application of this GA based heuristic in the fitness function of the GA operator we calculate the fitness of the chromosome based on the demand and share of distributor, the revenue function in this application is as follows

$$revenue_{jsn} = 0.133 * demand_{jn} + 0.9434 * demand_{jn} * (1 - floor_{j,r} / 100) \qquad (4.6.11)$$

Hence in the above equation the movie are scheduled only on the basis of demand of movie in that week and the percentage share of the demand which exhibitor will take. The movie selected in this manner are allocated to different screens based on screen allotment heuristic, in which the movie having higher demand will allocated to the screen having higher capacity. But the week-screen slots which are committed slot will assign the committed movie in that slot irrespective of the demand of that movie. Obtaining the same results shows that the GA based heuristic is applicable directly to the real world. The results of the approach are shown in the appendix. These schedules match each other except a few minor differences. These results exhibit the application validity of the proposed approach to more complex application settings. The next step in this endeavor would be to show better efficacy of the proposed approach in settings where parameters examined in the simulation study are crucial.

Another promise of the algorithm is in *reduction of manual operations* involved in the current SilverScreener implementation approach. The proposed approach provides automation of extremely time consuming operations. In the implementation approach of Eliashberg, Swami, Weinberg and Wierenga, 2001 for generating the schedule AMPL software is used, which gives the output in numeric form. Then for making the schedule the number are replaced by the movie corresponding to that number. Then these movies are sorted on the basis of predicted revenue of the movie. Hence the schedule is generated in two stages: selection of movies and allocation of screens to the selected movies. The proposed model can perform both the actions simultaneously and gives the output in the desired form. Hence the proposed model can help the exhibitors in generating the schedule effectively.

4.9 Managerial Implications, Conclusion and Directions for Future Research

The above results shows that under certain settings of parameters such as screen capacity, number of committed movies, number of Type I movies the proposed heuristic may perform better than the 2-stage SilverScreener approach proposed by Eliashberg, Swami, Weinberg and Wierenga, 2001. However the exact settings of these parameters may not be available in every application. This suggests the importance of considering the screen capacities in the exhibitor's decision making. All the main factors, namely, the capacity levels, the number of committed movies, and the number of rapidly decaying movies, are significant. The research suggests some important policies for the exhibitors.

First, it is implied that the exhibitors should not allocate screens to movies based on their demand alone. Even if the movies with high opening strength has higher demand in the beginning, its allocation to the high capacity screens might lead to inferior schedules because of the contract terms specific to movie types. The proposed model does the selection of movies and allocation of the selected movies to the screens simultaneously, thus provides automation of the procedure. It has been showed that the model also works on the real world data. Hence, this model can be used to make the recommendation more effectively. A lot of enhancement has been carried out in the basic SilverScreener model to deal with the various emerging issues. But still there are certain issues that require to be dealt.

In the real world implementation most of the movies are first "weekly" movies for some time, after which they play in matinee shows. Sometimes specific genre movies are played in matinee. In the present approach if we have *n* screen theatre, each week we generate the recommendation for best *n* movies. In this process, some movies are discarded which could be good matinee movies. Hence there is need to find the way by which such movies which seems least attractive but could be good movies for matinee shows or evening shows can be recommended with the best *n* movies.

The demand of any movie in a given theatre depends upon the movie running in the nearby theatre, the location of the other theatres. The present approach considers a stand-alone theatre and docs not take in to account these interdependencies among the theatres.

CHAPTER 5

MICRO- SCHEDULING PROBLEM

5.1 Introduction

The schedule generated in macro scheduling gives the list of movies which should run in the coming week on different screens of the multiplex theatre. The movies which are recommended to run on the different screens of the theatre in any week are from different genre; hence have different demand pattern during a day. The demand is also different on week days and weekend days. The recommended schedule in the macro scheduling is on the basis of average weekly demand. Hence this macro schedule is needed to be adjusted according to the demand of movie with in a day. This micro scheduling helps exhibitors to plan that which movie he should play, in which time slot and on which screens. so that he/she can maximize the revenue and account for other factors such as customer satisfaction.

Micro-scheduling is done at the theatre level. The macro schedule generated by SilverScreener, which contains the list of movies to be run in the coming weeks on different screens of any given theatres is sent to Senior Manager of theatre central planning office. He forwards this schedule to different theatre manager, who decide on the basis of that schedule which movie will run in which time slot during a day.

5.2 Real World Issues in Micro Scheduling

There are a lot of issues which are dealt by the manager of any multiplex theatre in generating this micro schedule. The various issues are as follows:

1. *Floor capacity constraint*: The multiplex theatre has different floor sections with some floors having more than one screen. Hence the manager of the theatre would not like to start two movies on different screens of the same floor simultaneously so that crowd on that floor does not exceed the capacity of that floor. This is because theatre may also suffer loss of the revenue which would otherwise be earned from concession sells.

2. *Two big movies at the same time:* The manager of the theatre would not like to be run two big movies at the same time, as it will need more staff to be hired to deal with the volume of people arriving for same time showings. If additional staff is not hired and the lines are long, the people may become frustrated, which could affect their consumption patterns at this theatre; while they probably stay and watch a movie but this unpleasant wait may result in a decreased desire to return the theatre.

3. *Customer Satisfaction*: For the convenience of the customer the theatre would like to schedule the movies in such a fashion that whenever the customer arrives at the theatre there should be at least one movie to start with in 30 minutes so that the average waiting time for a customer to watch a movie can be minimized.

4. *Availability of demand data of movies:* For making the micro schedule it is necessary to know the exact demand for each movie at each time slot of the day. Presently the theatre management considers the average weekly demand of any movie as this is sufficient for the macro scheduling, but at the micro level we need to know the time dependent demand function for any movie.

5. *Multiple copies of a movie*: The movies which are in the consideration set of the micro scheduling may have multiple copies, for the theatre manager it will not be desirable to schedule these multiple copies running at approximately the same time.

6. *Staff availability constraint*: The movies should be scheduled in such a manner so that at any time slot total numbers of movies which are completing their run length do not exceed the staff available for cleaning the theatre. Otherwise manager will not be able to schedule the next movie unless the screening room is cleaned; this may result in loss of revenue.

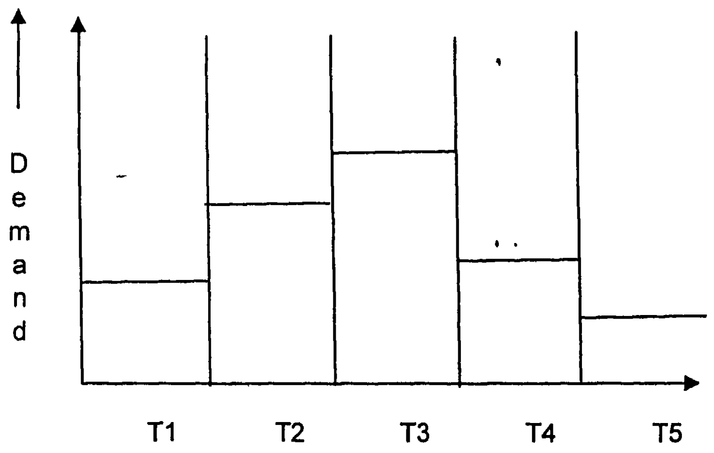
7. *Double booking of movies*: In case of the double booked movie, in which multiple copies of the same movie can run on different screen of the theatre at the same time, the demand of the movies will be interdependent. For example we have two copies of any given movies, and copy one has been started, than the demand of the second movie will demand on the time at which the movie one was started and at the screen on which it was scheduled. This dependency of demand makes the problem complex. The demands for the movies are also different on week days and week end, hence the schedule will also different for both of those periods.

5.3 Problem Formulation

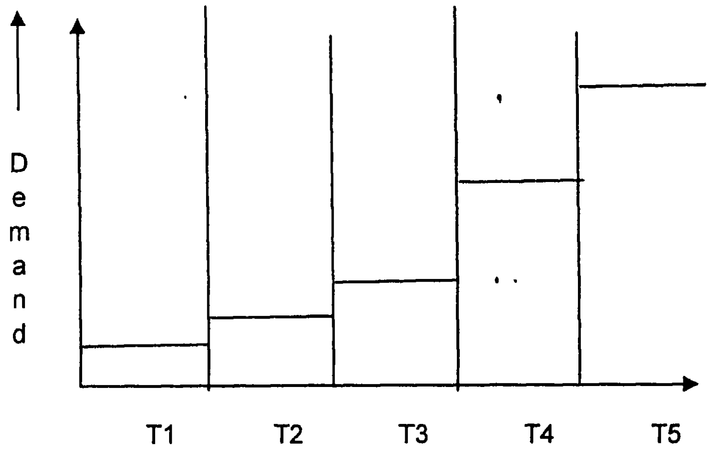
5.3.1 Calculation of Demand for Each Movie in different Time Slots

The movies which are selected for the screening have different genre like comedy, family, drama, horror, crime, action and others. These different types of movies have different demand pattern over the day. Hence in the given problem the movies are categorized according to the customer's preferences of time slots in which they prefer to watch the movies, like movies preferred by children should be schedule in the day time slots, romantic, action movies will be preferred to watch in the evening and night slots so these type of movies should be schedule in evening and night slots, mo. So the movie's demand varies according to class in which they are, during a day. For estimating the demand during different time slots the following procedure is followed.

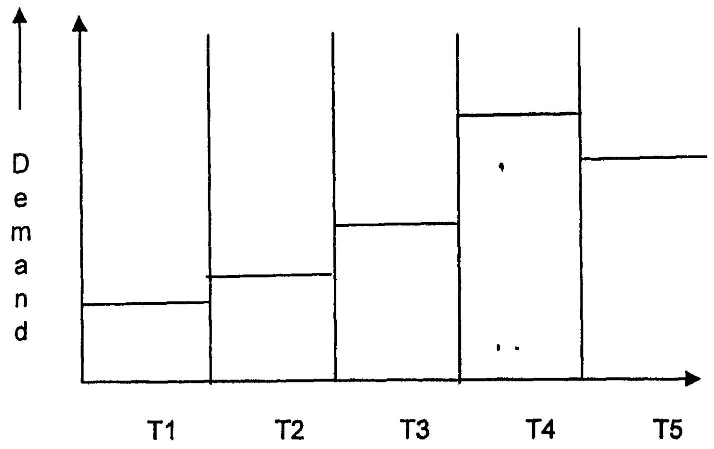
We are given the revenue forecast for each movie from the manager of the theater. From this revenue forecast we calculate the number of customers which will visit the theater during the week, by dividing this revenue forecast by the average price of the ticket. This will give the total number of customers which will visit to theater. This also has been observed that the movies have higher demand during the weekend days than the weekdays. So this whole demand of the week is further divided in to the weekend and weekdays, by multiplying by factor δ_1. From this demand we calculate the demand of a movie for any given day. This gives the potential demand of movie. Now the demand of movie varies with in a day, so this demand again multiplied by some other factor δ_2, which is different for different type of movies and in different time slots of a day. If some movie is preferred to be watch during the matinee shows it will have $\delta_2=0$ during the morning and evening slots, but have higher δ_2 during the matinee hours. We have divided the working hours of a theatre in five major time blocks (T1..T5) and are assuming that movies will have the same demand on all the times in any given major time block. The Figure 5.1 shows the demand pattern for five types of movies.



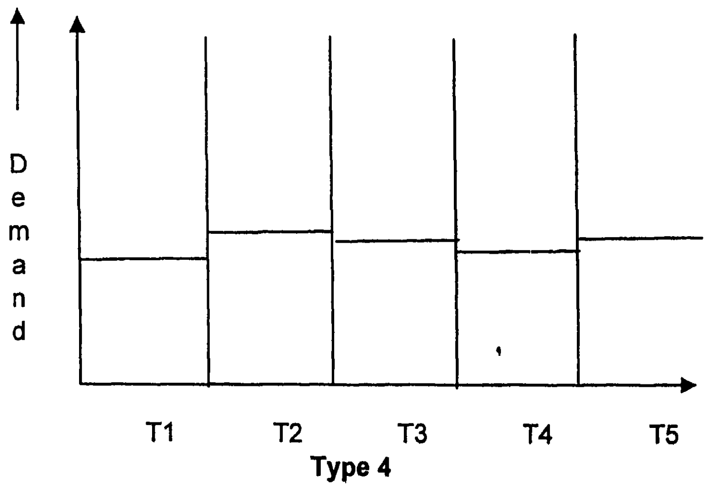
Type 1



Type 2



Type 3



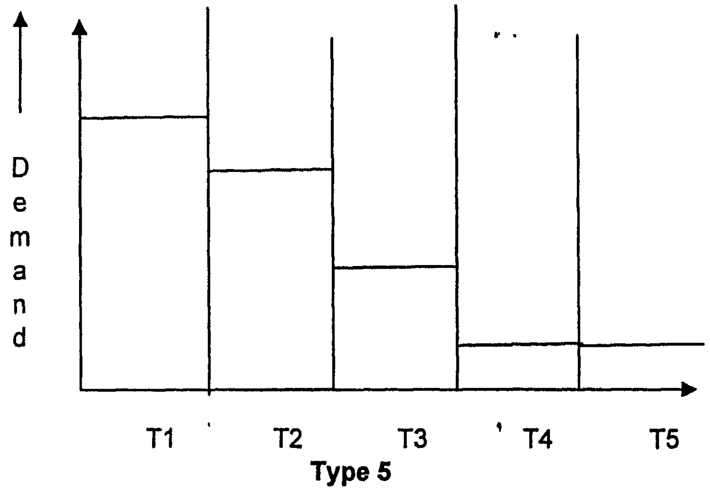


Figure 5.1: Demand Pattern for different Genre of Movies

Type 1 movies are the movies which have higher demand in the morning hours, but very low demand in the late hours. Children movies falls in this category. Type 2 movies have very high demand in matinee hours, these type of movies are family drama type movies. Type 3 movies have higher demand in evening hours and significant demand in night hours also. Type 4 movies have the high demand in night hours. Crime, action movies comes in this category. The type 5 movies have the average demand over the day. These are the movies which are preferred by every class of customer. Comedy movies fall in this category.

The ticket price is also different for weekend and weekdays, so the schedule will be different for both the cases. If the movies are scheduled in any given slot the movie will earn the revenue according to the demand in that time slot, but at the same time the variable costs like the salary of staff or electricity will also be incurred. The total revenue earned from different shows of a given movie on given screen can not exceed the potential revenue of the movie. If some movie is scheduled on some time slot it will continue till its end.

5.3.2 The Model

5.3.2.1 Model 1

Assumptions:

In order to simplify the exposition, we make the following assumption:

1. We assume that all the movies have single copy; hence no movie can be run simultaneously shown on two screens.
2. We assume that the movies which are to be run are screen specified by the macro schedule, now we have to only decide that in which time slot movie is to be scheduled.
3. Two big movies can start at the same time.
4. There is no loss in attendance or increase in variable cost from scheduling two movies at the same time, as opposed to say fifteen minutes apart.

Definition of Variables

T	- length of planning horizon (total number of time blocks),
S	- total number of screen in a theater,

$Visitor_{jwt}$	– demand for movie j, on day w, in time slot t,
x_{jwts}	- binary (decision) 0-1 variable which will take value 1 if movie j is scheduled on day w, in time slot t, and on screen s.
p_{wt}	- ticket price on day w in time slot t,
vc_s	. variable cost of scheduling any movie on screen s,
l_j	- run length of movie j,
c_s	- Cleaning time of screen s,
A	- time for ads, trailer etc.
δ_{cs}	- penalty associated with customer satisfaction,
$COMMIT$	– set of committed movies
$START_s$	- start time or the time slot in which the screen s will be available for first movie show
END_s	- the time slot after which no movie can run on screen s,
F_s	- set of screens which are on the same floor.
δ	- Average concession sell generated from customers.
rev_{jw}	- potential demand of movie j on day w

In the above definition of variables the planning horizon T is computed by dividing the difference of close and open time of the theatre. The block may be of 5,10,15 or 20 minutes.

Objective Function

The objective of the problem is to maximize the revenue earned from the sales of the ticket, and concession sales at the same time to minimize the variable cost of scheduling a movie and the penalty associated with customer satisfaction, which can be formulated as:

$$\sum_j\sum_w\sum_t\sum_s (Visitor_{jwt} * (p_{wt} + \delta) - vc_s) * x_{jwts} - \sum_j\sum_w\sum_t\sum_s \delta_{cs} * (x_{jwts} - \sum_{k\neq j}\sum_{t'=t}^{t+3}\sum_s x_{kwt's}) \quad \cdots(1)$$

In the above equation if movie j is scheduled at time slot t, on screen j than the binary variable will take the value 1. If no movie is scheduled after 3 time slots than the sum of all the binary variable will take the value zero and the objective function value will be reduced by the customer satisfaction cost penalty.

Constraint 1

The condition required for the feasible schedule is that the total number of movie running in any time slot at nay day should be less than the number of screens.

$$\sum_{j}\sum_{s} x_{jwts} \leq S \qquad \forall t \qquad \cdots(2)$$

At any given time slot if some movie is scheduled the binary value will take the value 1, since at any time block total number of movies to be start can not exceed the number of available screens. Hence the sum of all such variable should be less than equal to the number of screens.

Constraint 2

Sufficient time should be provided for cleaning, ads and trailer between two shows of a movie on the same screen on any given day. If some movie *j* is start at time block *t*, for ensuring that no other movies is scheduled during its run length, after completing its run length when the screen room is being cleaned, the sum of all other binary variable during that period should take the value zero. Hence the product of that binary variable and sum of all those variable will be zero. If the movie *j* is not scheduled in that time slot than any other movie can be scheduled in this case the product will be zero.

$$(\sum_{k \neq j} \sum_{t'=t}^{t+l_j+c_s} x_{kwt's}) * x_{jwts} = 0 \qquad \forall j, w, t, s \qquad \cdots(3)$$

Constraint 3

Two consecutive movies on the same floor should be at least half an hour apart, so that the crowd on any floor can not exceed the floor capacity.

$$x_{jwts} * \sum_{t'=t}^{t+3} x_{kwt's'} = 0 \quad \forall j, w, t, s, k \neq j, F_{s'} = F_s \qquad \cdots(4)$$

In the above equation if movie j is scheduled on screen s, than no other movie can be schedule on the screen which is on the same floor. Here we are assuming that the one time block is of 10 minutes. The sum of all other variable on the screens at the same floor will be zero. Hence the product will also be zero.

Constraint 4

Total revenue earned from different shows of given movie on any day can not exceed the potential of the movie

$$\sum_{t} x_{jwts} * Visitor_{jwt} * p_{wt} \leq rev_{jw} \qquad \forall j, \forall w \qquad \cdots(5)$$

Constraint 5

The movie should not split that is if movie is scheduled in any time slot it will continue till its end no other movie can be schedule during its running on that screen.

This constraint will take care by constraint 2.

Constraint 6

Movie can be scheduled in any time slot if it is in committed movie set irrespective of the demand of movie in that slot.

$$X_{jwts} = 1 \qquad j \in COMMIT \qquad \ldots (6)$$

Constraint 7

Movie can not be scheduled before the start time and end time of any screen of the theater.

$$X_{jwts} = 0 \qquad \forall j, w, s, t < START_s \text{ or } t > END_s - l_j \qquad \ldots (7)$$

5.3.2.2 An Alternative Formulation (Model 2)

Several approaches are currently under development which model the micro scheduling problem as an off-shoot of macro SilverScreener algorithms (Swami, Eliashberg and Weinberg 1999, Miller 2002). In this model, the problem is formulated as integer linear program.

Assumptions:

1. No movie is simultaneously shown on two screens.
2. Movies which are being considered for showing have already been chosen.
3. It takes the same amount of time to clean a screen and get it ready for the next movie regardless of the next movie is; there is no extra time required to get the screen ready if we switch the movie.
4. There is no increase in variable cost in switching.
5. There is no loss in attendance or increase in variable cost from scheduling two movies at the same time, as opposed to say fifteen minutes apart.

Definition of Variables

H	Number of screens in the theatre
C_c	Capacity of screens c c ε { 1,..........,H}
N	Number of candidate movies
Open	Time the theatre opens for showing movies
Close	Time the theatre closes for showing movies
NumBlocks	$\dfrac{Close - Open}{Block}$
rt_j'	Number of blocks for run time of movie j
rt_j	Extended number of blocks for run-time of movie j
$s_{l,j,t,u}$	Binary(decision) 0-1 variable which will take value 1 if movie j is scheduled on screen l at block t and the screen is first ready to be used again at block u, and 0 otherwise.

Equations of Constraint

Constraint 1:

If we start showing movie j on screen l at time t, it takes exactly rt_j blocks to show the movie and have the screen ready for showing the next movie, hence

$$\forall l, \forall j, \quad s_{l,j,t,u} = 0 \quad if \quad u \neq t + rt_j \qquad \cdots(1)$$

Constraint 2:

The movie can not be started before the theatre opens, nor can the movie running after the theatre closes. Thus

$$\forall l, \forall j, \quad s_{l,j,t,rt} = 0 \ \ if \ \ t < Open, \ \ t+rt_j > Close \qquad \cdots(2)$$

Constraint 3:

At any given time block *T*, on any screen only one movie can be shown. For movie *j*, if it starts at or before time *T-rt_j*, then it is done running by time *T*, and the screen is ready of for use again. If it starts running between time $T - rt_j + 1$ and *T*, then it will be running at time *T*. Thus,

$$\forall l, \forall T, \quad \sum_{j=1}^{N} \sum_{t=T-rt_j+1}^{T} s_{l,j,t,t+rt_j} \le 1 \qquad \cdots(3)$$

Constraint 4:

For any movie *j*, it is only being shown on one screen at a given time *T*. If we start movie *j* before or at time $T - rt_j$, then that run will not be showing at time *T*. thus to be showing at time *T* we must have started the movie between time $T - rt_j + 1$ and *T*. Thus we led to

$$\forall j, \forall T, \quad \sum_{l=1}^{H} \sum_{t=T-rt_j+1}^{T} s_{l,j,t,t+rt_j} \le 1 \qquad \cdots(4)$$

Comparison of Model 1 and Model 2 :

The formulation for micro scheduling problem in Model 1, is based on the assumption that the movie which are to be scheduled has already been specified that on which screen it will run by the macro scheduling, only the time slots is to be determined. The customer satisfaction has been taken care by assigning the penalty to objective function. Hence the objective function of the formulation consists of the revenue earned from the sell of ticket, concession sells, it also consists of the variable cost of assigning the movie to any screen and the penalty of deviating from customer satisfaction condition.

The constraints handled by the alternative approach, Model 2 has been taken care by Model 1 but in different way i.e. constraint (2) and (5) of Model 1 which were for ensuring that the movie will complete its run length and will not assign any other movie at that screen during its run length and enough time will be given for cleaning after its completing run length,

$$\left(\sum_{k\neq j} \sum_{t'=t}^{t+l_j+c_s} x_{kwt's}\right) * x_{jwts} \le 1 \qquad \forall\, j, w, t, s$$

this been handled by eq (3) in Model 2

$$\forall l, \forall T, \qquad \sum_{j=1}^{N} \sum_{t=t'-rt_j+1}^{t'} S_{l,t,j,t-rt_j} \quad \le 1$$

Similarly the end and start constraint have been handled in both formulation but in different ways. Model 1 proposes constraint (3), which is floor constraint in which if one movie is assign on first floor the next movie on that floor will come after at least half an hour (assuming that each block is of 10 min).

$$x_{jwts} * \sum_{t'=t}^{t+3} x_{kwt's'} \quad = 0 \quad \forall j, w, t, s, k \ne j, F_{s'} = F_s$$

In addition to this in Model 1, for restricting the number of shows in a day, constraint (4) ensures that the total revenue earned from the movie can not exceed the potential demand of the movie in any given day, i.e.

$$\sum_{t} x_{jwts} * Visitor_{jwt} * p_{wt} \quad \le rev_{jw} \qquad \forall j, \forall w \qquad \cdots(5)$$

5.3.2.3 Selected Model (Model 3):

We combine the merits of the two approaches and finally selected the following model. We make the following assumption for tractability of the initial analysis.

Assumptions:

1. We assume that all the movies have single copy.
2. We do not consider multiple booking of the movies.
3. The list of movies which are in the consideration set of micro scheduling may be different from the recommended list of macro scheduling. Because there may be the case when some movie in the macro planning consideration set which was not selected due to having lower weekly predicted demand, may have higher demand in particular time slots of the day than the movies, recommended by macro planning.
4. We assume that the floor of the multiplex theatre has enough capacity; hence more than one movie can be assigned on any given floor.
5. The time required to clean for all screen is same.

6. There is no increase in variable cost in switching the movie from one screen to next screen, and the time required for switching the movies from one screen to another is dominated by the time given for cleaning, ads and trailers between two consecutive shows on any screen.

Definition of Variables

T	- length of planning horizon,
S	- total no of screen in a theater,
$visitor_{j,t}$	– demand for movie j, in time slot t,
x_{jts}	- binary (decision) 0-1 variable which will take value 1 if movie j is scheduled in time slot t, and on screen s.
p_t	- ticket price in time slot t,
vc_s	- variable cost of scheduling movie on screen s,
l_j	- run length of movie j,
c_s	- Cleaning time of screen s,
$start_s$	- start time or the time slot in which the screen s will be available for first movie show
end_s	- the time slot after which no movie can run on screen s,
$staff$	- the total number of cleaning staff available at any time block t.
δ_{cs}	- the penalty for customer dissatisfaction

Objective Function:

The objective of the problem is to maximize the revenue earned from the sales of the ticket, and concession sales at the same time to minimize the variable cost of scheduling a movie, which can be formulated as:

$$\sum_j \sum_s \sum_t (\min(visitor_{j,t}, cap_s) * p_t * x_{j,s,t}) - vc_s * x_{j,s,t}) \qquad \cdots(1)$$

The above function tries to schedule the movie at each third time block, if no movie is scheduled penalty equal to customer satisfaction cost is added in the objective function. Hence the objective function tries to minimize these penalties.

Constraint 1:

This constraint will ensure that at any time block t and any given screen s only one movie can be run. This can be ensured by not assigning any other movie during its run length. If any movie j starts at or before t-l_j, than it is done running at time t. But if any movie is scheduled during its run length it will make the binary variable start 1 so the sum of the entire start variable should be less than or equal to 1.

$$\forall s, \forall t, \quad \sum_{j=1}^{N} \sum_{t1=t-l_j+1}^{t} x_{j,s,t1} \le 1 \qquad \cdots(2)$$

In the above equation in the second sum the lower bound will depend on the the movie.

Constraint 2:
This constraint will ensure that the same movie can not be run on more than one screen at the same time. Similar to constraint 1 for movie running at time block t it should be scheduled during time block t-$l[j]+1$ to t. The sum of all the start time variable over all the screen should be less than one.

$$\forall j, \forall t, \quad \sum_{s=1}^{S} \sum_{t1=t-l_j+1}^{t} x_{j,s,t1} \le 1 \qquad \cdots(3)$$

in this equation also the lower bound depends on the run length of movie which has been scheduled.

Constraint 3:

The movie can not be started before the theatre opens and the movie should be completed its run length before the end time of any screen.

$$\forall j, \forall s, \quad x_{j,s,t} = 0 \quad if \quad t < start[s] \qquad \cdots(4)$$

$$x_{j,s,t} = 0 \quad if \quad t > end[s] \qquad \cdots(5)$$

Constraint 4:
The movie should be scheduled in such a way that when it completes its run length, there should be no delay in assigning next movie on that screen due to unavailability of staff. For completing its run length at any time block t, it should be scheduled at t-l_j, hence at t-l_j, the sum of start variable should be less than staff available.

$$\forall t, \quad \sum_{j=1}^{N} \sum_{s=1}^{S} x_{j,s,t-l_j} \le staff \quad such\,that \; t - l_j \ge 0$$

5.4 Input Needed for Micro Scheduling

Below are the input information needed to for generating the micro schedule:

1. Number of screens S, its capacity cap_s, and its variable cost vc_s (cost for running the single show of any movie).
2. Each day, list of candidate movies (which can be shown): N
3. Run-time for each movie l_j, and time to clean the screen c_s.
4. Length of each time block B (measured in minutes).
5. Open and close time for each screen.
6. Potential demand of movie which it can earn in a day and also the demand at each time block.
7. The number of staff available at any time block of the planning horizon for cleaning the screen.

5.5 Testing of Model

To check whether the given model for micro schedule gives the feasible schedule, we applied our model on the simulated data used by Jedidi, Krider and Weinberg (1998) for their study on *clustering at the movies*. Different scenarios were generated and the model was applied on the problems of different scenario. The following is the methodology of data setup.

5.5.1 Data Setup :

Micro schedule uses the movies recommended by macro schedule. We run the macro plan on the data used by Jedidi, Krider and Weinberg (1998). This gave the list of movies to be considered for the micro scheduling. The movies potential demand for the week was noted from the input file to macro plan. To calculate the potential demand of the considered for a day we multiplied this demand by factor δ_1 (0.7 for week-end and 0.3 for week days). Now to calculate the demand of movies in each time block we multiplied this one day potential demand by the factor δ_2, this factor depend on the genre of movies. As shown in figure 5.1, we divide the whole planning horizon in to five major time blocks, and assumed that the demand of movie at all the time blocks, falling in that major time block will be same. The factor δ_2 can take any one value out of the five assumed values depending on the genre of the movie and time slots at which we want to calculate the demand. The five values of δ_2, which we assumed are 0.7, 0.6, 0.4, 0.2 and 0.1. To decide the genre of the movies we used the relative proportion of different genre of Type 1 (Family, Drama movies, which are preferred in matinee shows) 28%, Type 2 (Horror movies, which are preferred in night shows) 10%, Type 3 (Crime and action movies, which are preferred in evening shows) 28%, Type 4 (Comedy movies, which have almost the same demand) 31% and Type 5 (Children movies and others which have generally higher demand in the morning slots) 4% (Jedidi, Krider and Weinberg, 1998). Now if the movie is of Type 1 genre its potential demand for the day will be multiplied by 0.7 in the matinee hours and by 0.6 in evening hours and so on. Hence the sequence of multiplication factor will change according to the genre of movies. For other parameter of the micro scheduling such as screen capacity we randomly assign the some value to each screen and for variable cost we assumed that variable cost of any screen be ten times the capacity of that screen. We also randomly assigned the run length of different movies varying between 5 to 9 time blocks (each time block is of fifteen minutes), which the Hollywood movies generally have.

5.5.2 Problem Design:

For any theatre manager the factors which may be of interest in case of micro scheduling are capacity utilization and staff availability which we have highlighted in our study. For studying the sensitivity of these parameter (staff availability) or on these parameter (capacity utilization), we generate the schedule for various scenario. For a given set of movies from macro schedule we generate four types of scenario by varying the staff available (1 or 2) and by varying the level of potential demand for a day (week day or week end days). Six sets of four such cases (24 scenarios) were generated. The capacity utilization for a given scenario was calculated in the following way:

$$\text{Total Capacity Available } (A) = (T * Cap_1 + T * Cap_2 \dots\dots\dots + T * Cap_n)$$

$$\text{Unutilized capacity } (B) = (T_1' * Cap1 + T_1' * Cap1 \dots\dots + T_1' * Cap1)$$

Table 5.1: Input Scenarios for Micro Scheduling

Scenario	Staff	Week-end/week days[*]	Cum. Revenue	% of high quality movies	Avg. Age of Movies
1.1	1	0	10097.1	66.6	5
1.2	1	1	23559.6	66.6	5
1.3	2	0	10097.1	66.6	5
1.4	2	1	23559.6	66.6	5
2.1	1	0	13525.2	50	2.5
2.2	1	1	31558.8	50	2.5
2.3	2	0	13525.2	50	2.5
2.4	2	1	31558.8	50	2.5
3.1	1	0	15634.5	33.3	2.84
3.2	1	1	36480.5	33.3	2.84
3.3	2	0	15634.5	33.3	2.84
3.4	2	1	36480.5	33.3	2.84
4.1	1	0	9561.3	66.6	5
4.2	1	1	22309.7	66.6	5
4.3	2	0	9561.3	66.6	5
4.4	2	1	22309.7	66.6	5
5.1	1	0	16531.5	83.3	2.5
5.2	1	1	38573.5	83.3	2.5
5.3	2	0	16531.5	83.3	2.5
5.4	2	1	38573.5	83.3	2.5
6.1	1	0	18102.3	50	2.84
6.2	1	1	42238.7	50	2.84
6.3	2	0	18102.3	50	2.84
6.4	2	1	42238.7	50	2.84

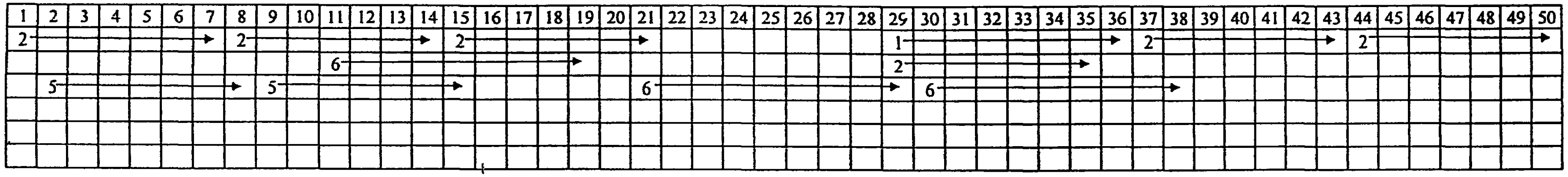
5.5.3 Results

For all the above scenarios the corresponding micro schedules were generated. The schedules are shown in Table 5.2. The summary of the results are shown in Table 5.3. Table summarizes the results on the basis of different movies scheduled out of the considered movie, maximum number of repetition, average number of repeated shows and capacity utilization.

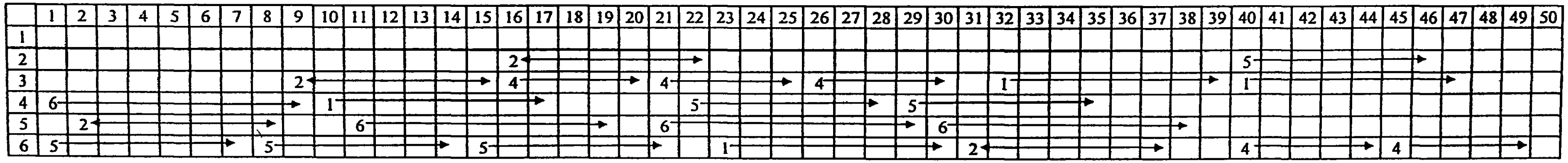
[*] 0 represents the week days while 1 represents the week end days

Table 5.2 Micro Schedules for Various Scenarios

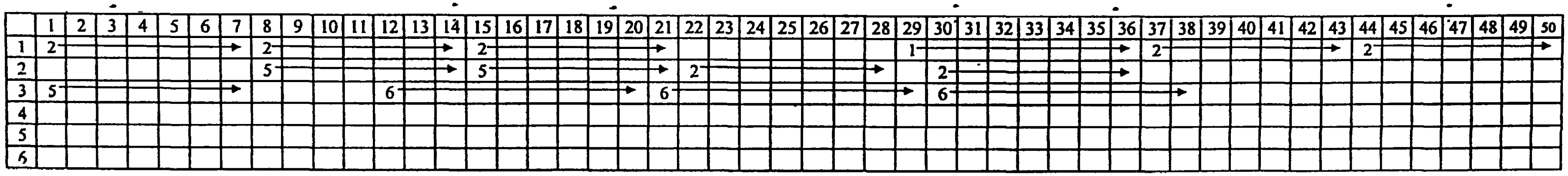
Scenario 1.1 (week2, week days, staff 1)



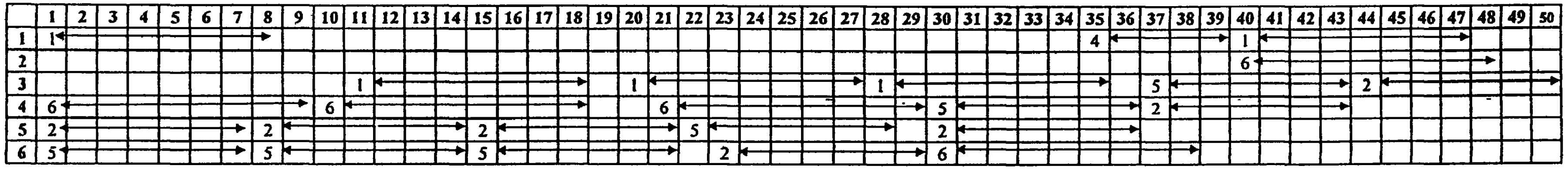
Scenario 1.2 (week2, week-end days, staff 1)



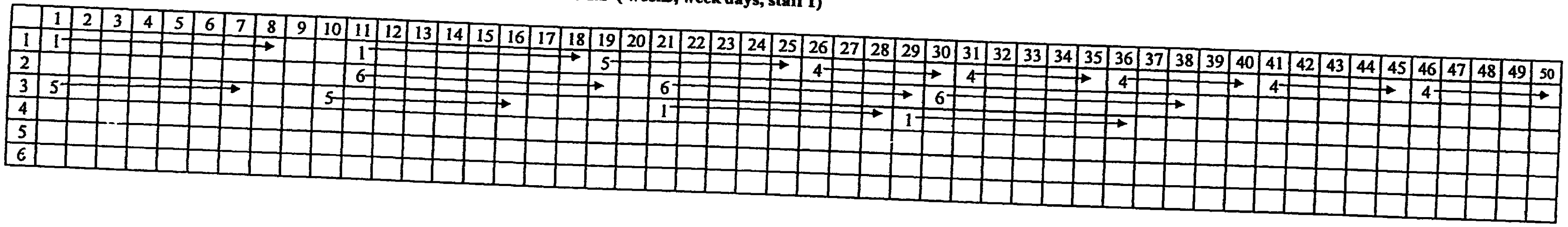
Scenario 1.3 (week2, week days, staff 2)



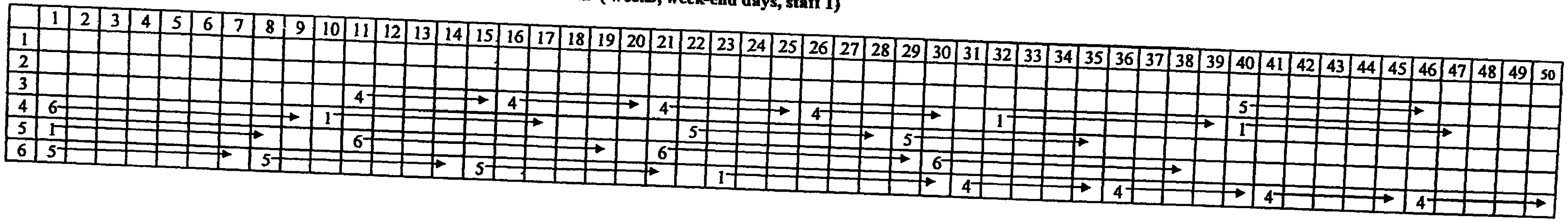
Scenario 1.4 (week2, week-end days, staff 2)



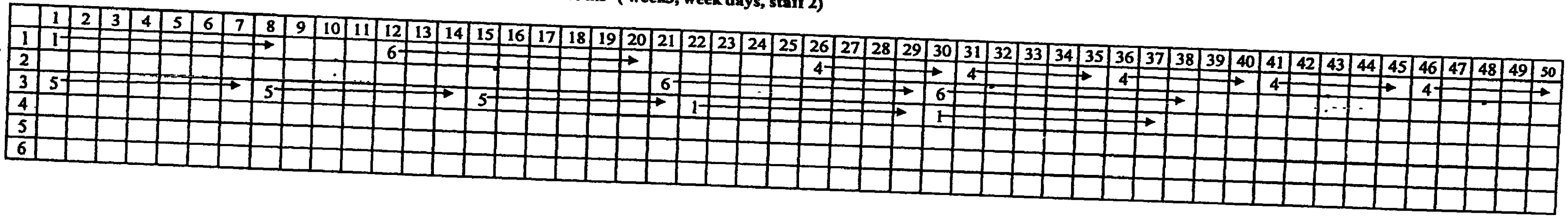
Scenario 2.1 (week3, week days, staff 1)



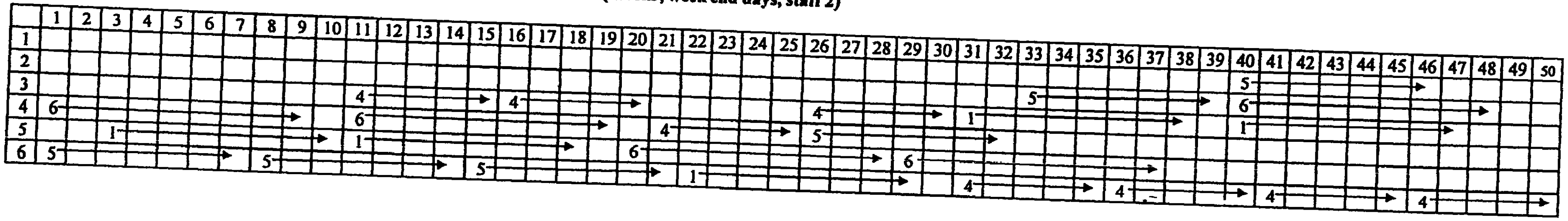
Scenario 2.2 (week3, week-end days, staff 1)



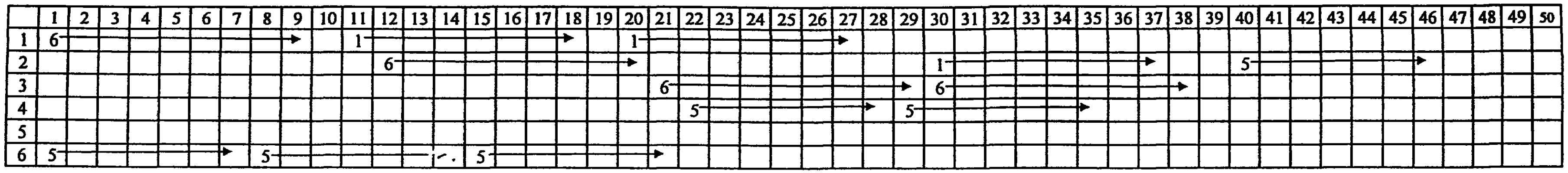
Scenario 2.3 (week3, week days, staff 2)



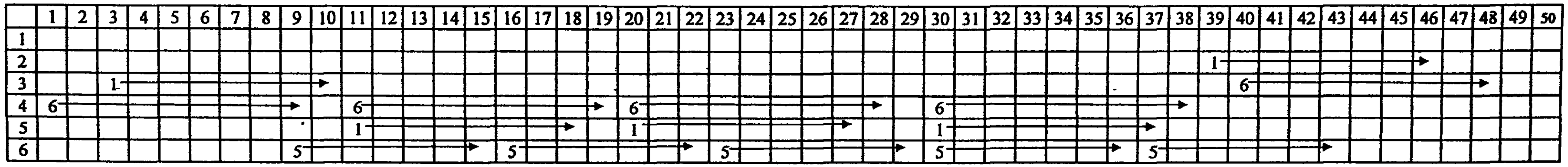
Scenario 2.4 (week3, week end days, staff 2)



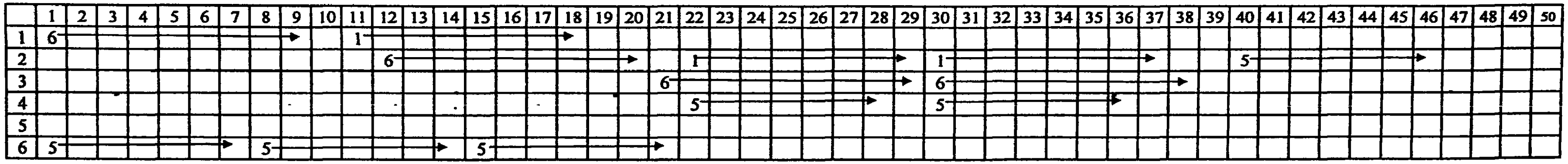
Scenario 3.1 (week4, week days, staff 1)



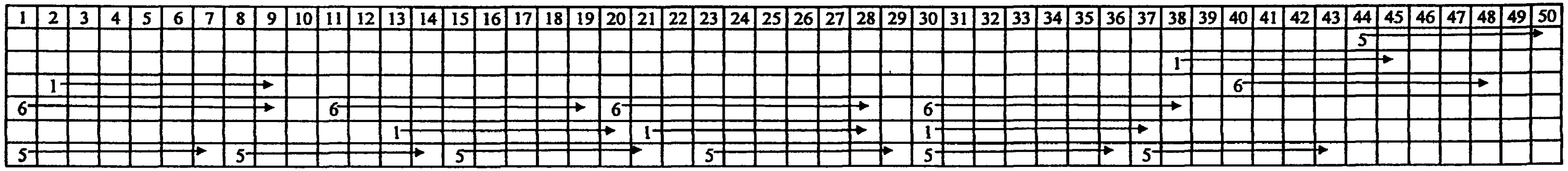
Scenario 3.2 (week4, week-end days, staff 1)



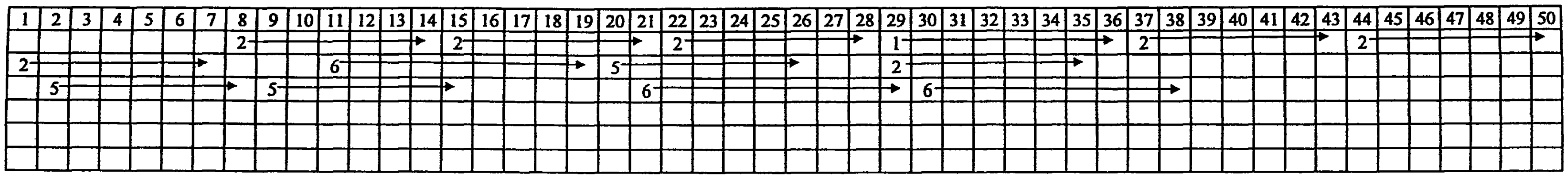
Scenario 3.3 (week4, week days, staff 2)



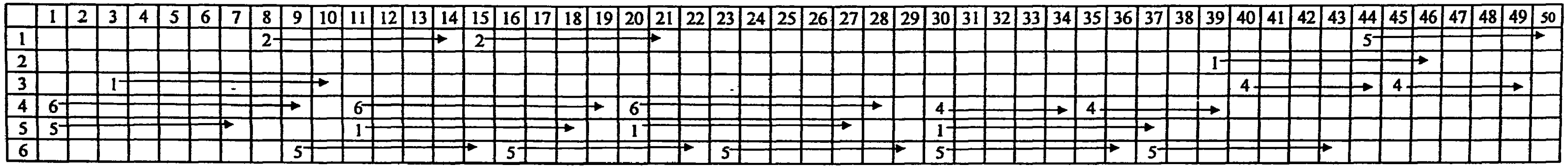
Scenario 3.4 (week4, week-end days, staff 2)



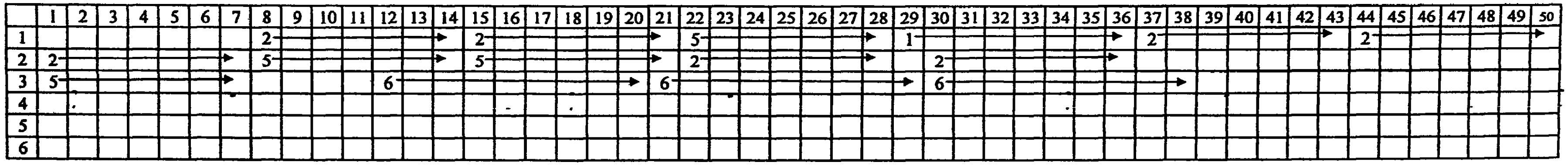
Scenario 4.1 (week2, week days, staff 1)



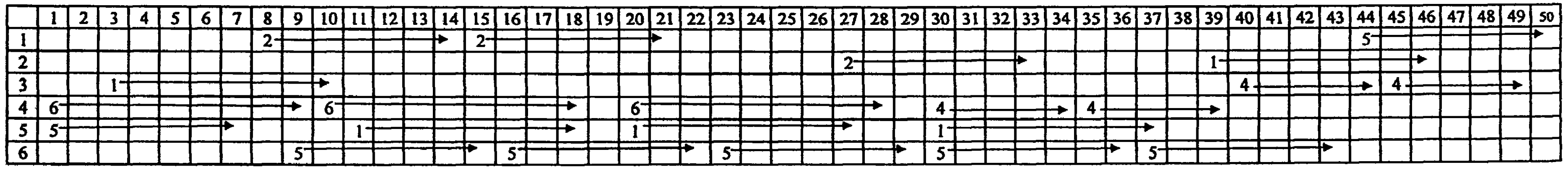
Scenario 4.2 (week2, week-end days, staff 1)



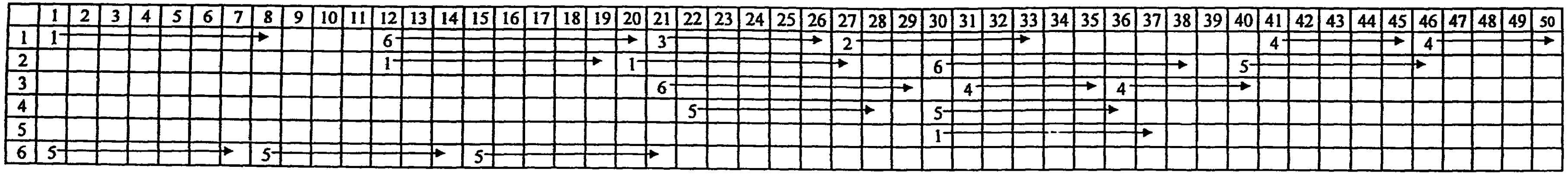
Scenario 4.3 (week2, week days, staff 2)



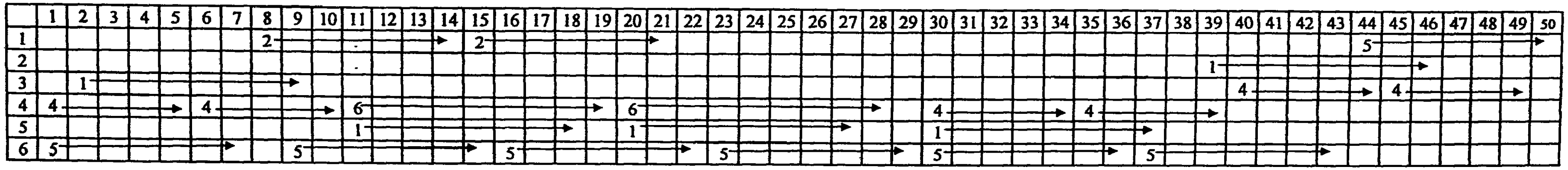
Scenario 4.4 (week2, week-end days, staff 2)



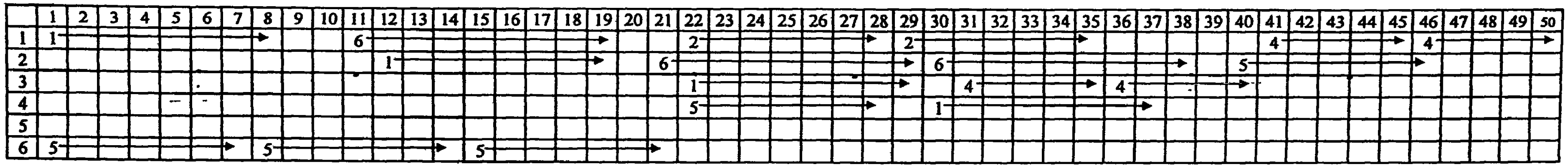
Scenario 5.1 (week3, week days, staff 1)



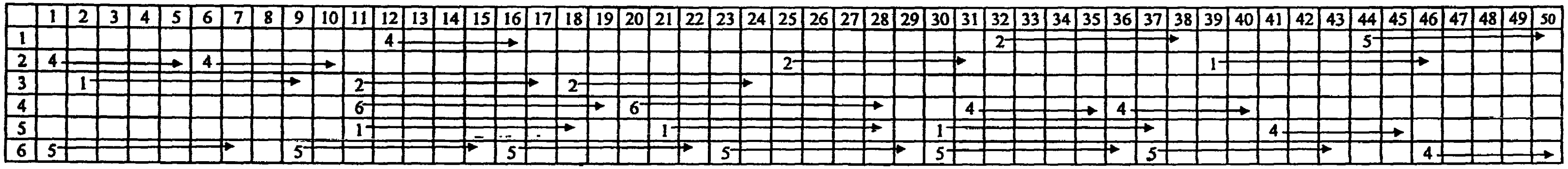
Scenario 5.2 (week3, week-end days, staff 1)



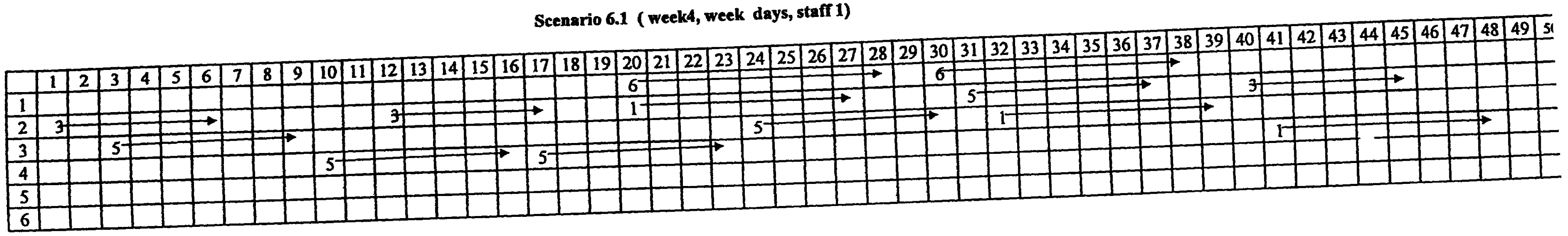
Scenario 5.3 (week3, week days, staff 2)



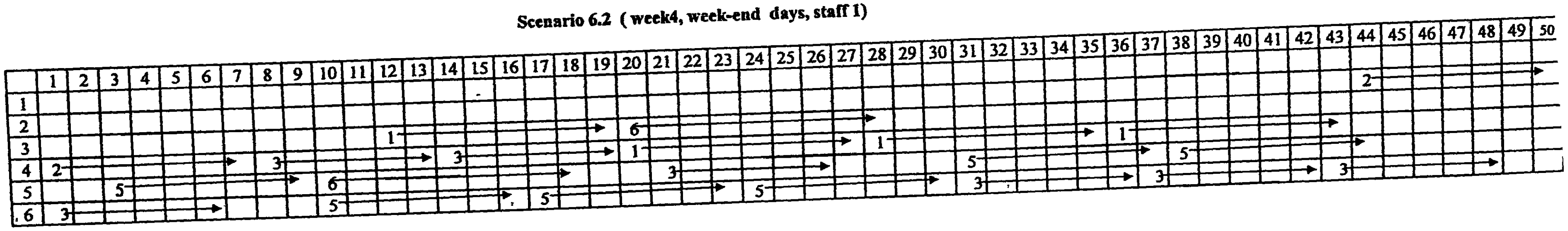
Scenario 5.4 (week3, week-end days, staff 2)



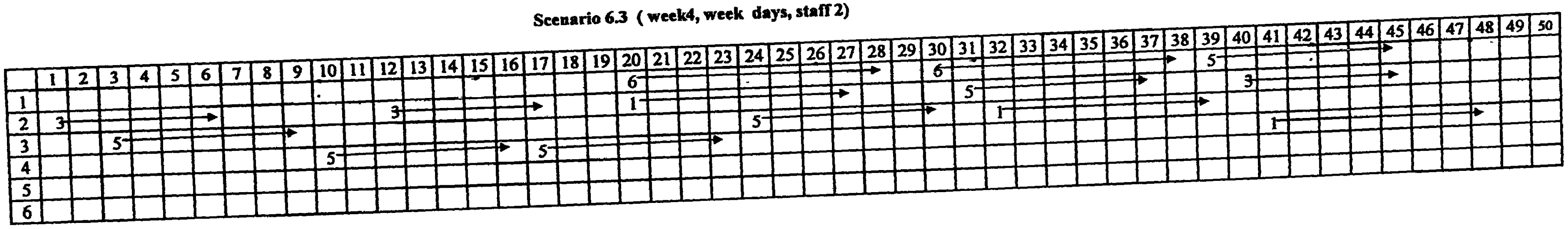
Scenario 6.1 (week4, week days, staff 1)



Scenario 6.2 (week4, week-end days, staff 1)



Scenario 6.3 (week4, week days, staff 2)



Scenario 6.4 (week4, week-end days, staff 2)

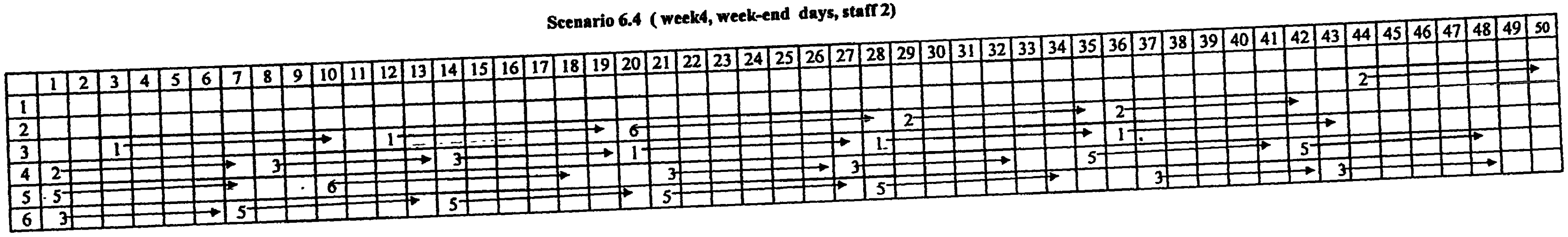


Table 5.3: Output Table of Different Scenarios

Scenario	Different No. Of Movies Shown	Max No. of Repetition	Average No. of Repeated Shows	Capacity Utilization	
				High	Low
1.1	4	6	3	0	21.88
1.2	5	6	4.6	48.23	10.34
1.3	4	7	3.5	0	25.56
1.4	5	7	2.8	43.55	17.33
2.1	4	5	3.75	0	24.81
2.2	4	8	5.75	49.7	11.97
2.3	4	5	3.5	0	23.64
2.4	4	8	6.25	14.27	48.46
3.1	3	6	4.3	14.35	15.84
3.2	3	5	5	40.074	6.74
3.3	3	6	4.3	15.78	16.78
3.4	3	7	5.66	43.68	8.28
4.1	4	6	3.5	0	25.52
4.2	5	7	5	42.81	11.68
4.3	4	7	3.75	0	27.36
4.4	4	7	5.2	46.23	9.56
5.1	6	6	3.16	13.642	21.76
5.2	5	7	4.4	43.56	12.01
5.3	5	5	3.6	16.15	19.89
5.4	5	6	5	49.23	13.23
6.1	4	5	3.25	3.23	39.25
6.2	5	7	5.2	41.46	4.89
6.3	4	6	3.75	4.23	42.61
6.4	5	7	5	48.25	7.58

5.5.1 Discussion

In week days the screens of low capacity/ low variable costs are highly used, while in week-end days the screens having high capacity/ high variable costs are used. The results show that when the average of the movies considered in various scenarios is lower then the screens of higher capacity are being utilized more. This is because the lower average of the movies represents that there are more number of new releases for which the potential demand is high.

The average number of shows in week-end days are higher than in week days, this difference in the number of shows are filled by the movies, which have the shorter run length. Hence the movies which have shorter run length are preferred than the movies which have longer run length. Because those movies can be schedule more frequently.

The usefulness of staff availability is higher in week days than in week-end days. The potential demands of movies are lower in week days than in week-end days. A movie on any screen can be scheduled only when it justifies the operating cost of screen and other running costs (as assumed in input data). In case of week days there are very few movies which justifies these costs. Hence the movies are scheduled in that time slots where these have the higher demand. If there is any delay due to unavailability of staff it will has to go the screens which has the lower capacity, this may result in the decrease in capacity utilization. But in case of week-end days a lager number of movies are available which can justify the costs which incurred in running that movie. If the movie can not be scheduled in that time slot where it has the higher demand will replace by some other movie which is of different class of genre and has the demand high in the other slots.

5.6 Managerial Implications and Directions for Future research

The proposed model is helpful to management in taking the various important decisions. In micro scheduling for any theatre manager the most important factor is capacity utilization. Because as the capacity utilization increases the revenue which he can earn increases.

The movies which have shorter run length are preferred than the movies which have longer run length. Because those movies can be schedule more frequently This may be useful in increasing the usefulness of the macro schedule, in macro scheduling the movies are selected only on the basis of contract term and predicted demand of the movies in that week. There may be the case when the movies which have shorter run length, have lower estimated demand in that week, the macro plan for recommending the movie will not recommend that movie. Hence to generate the best recommendation the run length can also be the significant factor.

This current research provides the basic idea of the problem. Still there are a lot of issues which are not dealt by the current model. Like in micro scheduling the customer satisfaction is also one of the important factors. The current model can be extended to include similar issues such as floor capacity constraint or double booking of movies.

CHAPTER 6

CONCLUSION AND LIMITATIONS

In this thesis we address the general retail space allocation problem. This problem was studied in the context of motion picture exhibitors. We solve the exhibitor's problem in two stages. At macro level we applied the modified GA based algorithm to solve the real world problem. Results match with the best implementation approach available so far. In this thesis we also show an improvement of the proposed approach over SilverScreener approach by 23%. We have tested our model for 26 movies, 14 screens, and 8 weeks problems usually tested in real world. Hence this model can be used to make realistic recommendation quite effectively.

The benefits of the proposed approach are revealed at several levels. One is *better selection of movies*. This is choosing a subset of movies out of those considered and is a direct output of the proposed algorithm. Another direct output is *better scheduling of movies* in terms of choosing their appropriate run lengths. The third function perform by the proposed algorithm is *better capacity allocation to movies*. This is the benefit that reveals itself significantly in the simulation study in conjunction with the other factors.

At micro level, the proposed model is intended to help the exhibitor in utilizing theatre capacity in a better way. The problem of micro scheduling matches the problem of parallel machine scheduling (m machine, n jobs) and time tabling problem. In the current thesis we have given the Integer Linear formulation of the problem and solve the model for the simulated data.

The results at the micro level shows that different within day schedules can be generated for different settings of input parameters. The schedules are appropriately generated for input length of movies, screens and time slots. A desirable feature of the programming is that the nature of schedules generated differs for week end and week days programming. This is to be expected as the demand of the movies is higher on week end days than on the week days. Therefore on week end days the capacity utilization of big screens are higher while on week days the lower capacity screens have higher capacity utilization.

utilization of big screens are higher while on week days the lower capacity screens have higher capacity utilization.

We now discuss some limitation of the current approach, which provide potential future research ideas. At macro programming level, the program can be extended to include stochastic demand pattern. Further some additional managerial preferences could be forced in to programming such as after commitment period, a movie is not allocated a higher capacity screen than that was allocated to it during commitment period.

At micro programming level, the current approach does not explicitly account for such managerial consideration such as incorporation of customer satisfaction in to objective function, constraint of floor capacity on simultaneous scheduling of movies on that floor and multiple bookings of the same movie on same day/ time slot. Further effects of programming on simulating queue length and their effects on theatre costs would provide useful managerial output. Additionally complexity analysis of the optimization problem would provide useful insights in to the efficiency aspects of the problem.

APPENDIX - 1

GA CODE FOR MACRO SCHEDULING PROBLEM

```
/*Code with bold fonts are for the case in which the screen allotment heuristic (For
comparison with SilverScreener approach) has been used.*/
/*MACRO SCHEDULING OF MOVIES CONSIDERING THE OBLIGATION PERIOD AND
COMMITMENT CONSTRAINT for MUNT 35-39 WEEK MOVIE PLANNING*/

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
#define NUM_MOVIES 27
#define NUM_WEEKS 5
#define NUM_SCR 13
#define MAXSLOT 65

struct chrom
        {
                int no;
                int b[NUM_WEEKS][NUM_SCR];
                float revenue;
                int copies;
                struct chrom *next;
        };
void selection();
void crossover();
void mutation();
void readdata();
float evaluation(int b[NUM_WEEKS][NUM_SCR]);
int rel_dt[NUM_MOVIES];
float demand[NUM_MOVIES][NUM_WEEKS],sort_scr[NUM_SCR],temp;
float contract_term[NUM_MOVIES][NUM_WEEKS];
static int commited[10];
int rrr;
char movie[NUM_MOVIES][20];
float screen_cap[NUM_SCR];
struct chrom *head,*h;
int popsize,maxgen,OPD[NUM_MOVIES],rlength,t,index_cap[NUM_SCR];
float pc,pm,tot_pop0,tot_pop1,tot_pc0,tot_pc1,tot_pm0,tot_pm1;
int commit[NUM_WEEKS][NUM_SCR],i,j,n,total;
int gen=0;
main()
{
 int x,y,z,k,xx,count,num1,num2,temp2;
 int ab,bc,cd,de,hpop,best_pop;
 float highest,hpc,hpm,tot_pop,tot_pc,tot_pm,best_pc,best_pm;
 static int best_revenue[3][3];
 int l,flag,flag1,flag2,flag3,flag4,flag7,flag8,m,r,temp1;
 struct chrom  *p,*q,*best,*h_temp;
 highest=0.0;hpop=0;

screen_cap[0]=99906;screen_cap[1]=99904;screen_cap[2]=152320;screen_cap[3]=51072;screen_
```

```
cap[4]=46592;screen_cap[5]=73026;screen_cap[6]=73024;screen_cap[7]=77504;screen_cap[8]=
79298;screen_cap[9]=79296;screen_cap[10]=171136;screen_cap[11]=49286;screen_cap[12]=448
00;
 for(i=0;i<NUM_SCR;i++)
 {
    sort_scr[i]=screen_cap[i];
     index_cap[i]=i;
 }

 for(i=0;i<NUM_SCR-1;i++)
 for(j=i;j<NUM_SCR;j++)
 {
     if(sort_scr[j]>sort_scr[i])
     {
          temp=sort_scr[j];
                 temp1=index_cap[j];
          sort_scr[j]=sort_scr[i];
          index_cap[j]=index_cap[i];
          sort_scr[i]=temp;
          index_cap[i]=temp1;
     }
 }
 readdata();

 printf("\nscreen capacity\n");
 for(i=0;i<NUM_SCR;i++)
 printf("%f      ",screen_cap[i]);

 for(y=0;y<NUM_WEEKS;y++)
 for(z=0;z<NUM_SCR;z++)
 commit[y][z]=99;

commit[0][2]=4;commit[0][9]=3;commit[1][7]=1;commit[1][1]=0;commit[2][1]=2;commit[3][2]
=5;
 i=0;
 for(y=0;y<NUM_WEEKS;y++) {
 for(z=0;z<NUM_SCR;z++){
 if(commit[y][z]!=99){commited[i]=commit[y][z]; i++;}}}
 total=i;
 printf("\nthe commited movies are %d",total);
 for(i=0;i<total;i++)
 printf("\n%d %d",i,commited[i]);

          for(ab=0;ab<1;ab++)
          for(bc=0;bc<1;bc++)
          for(cd=0;cd<1;cd++)
          for(de=0;de<1;de++)
          {
                  gen=0;
                  h= (struct chrom *)malloc(sizeof(struct chrom));
          if(h==NULL){printf("couldn't assign\n");}
                  head= h;
                  switch(ab)
                          {
                                  case 0:popsize=400;break;
                     case 1:popsize=500;break;
                  }
          switch(bc)
                  {
                          case 0:pc=0.95;break;
                          case 1:pc=0.99;break;
                  }
      switch(cd)
                  {
                          case 0:pm=0.01;break;
                          case 1:pm=0.03;break;
                  }

          switch(de)
                  {
                          case 0:rrr=1;break;
            case 1:rrr=25;break;
            case 2:rrr=100;break;
            case 3:rrr=400;break;
            case 4:rrr=6000;break;
                  }

          maxgen=2000;
          pc*=100;
          srand(rrr);
          pm*=100;
     x=0;
     do
     {
                  h->no=x;
                  h->copies=0;
                  for(y=0;y<NUM_WEEKS;y++)
                          for(z=0;z<NUM_SCR;z++)
                          {
                                  if(commit[y][z]!=99)
                                  h->b[y][z]=commit[y][z];
                                  else
                                  {
                                          flag=0;
                                          while(flag==0)
                                          {
                                                  flag1=0;flag2=1;
                                                  r=rand()% NUM_MOVIES;
                          //checking whether the generated movie is commited
  movie
                                                  for(i=0;i<total;i++)
                                                  if(r==commited[i]){flag2=0;
  break;}
                                                  if(flag2==0)
                                                  {

     for(i=0;i<NUM_WEEKS;i++)

     for(j=0;j<NUM_SCR;j++)
                                                          {
```

```
        if(r==commit[i][j]){temp1=i;break;}
                                                                        }
                                                   //loop for checking whether the generated movie which is
commited also is being assigned after commited period
                                                                          if(temp1>=y)flag=0;
                                                                          else
                                                                          {

        for(l=0;l<z;l++)if(r==h->b[y][l])flag1=1;

        if((rel_dt[r]<=y)&&(flag1==0))

                                                                                  {
                                                                                          h-
>b[y][z]=r;

                                                                                          flag=1;
                                                                                  }
                                                                          }
                                                                  }
                                                                  else
                                                                  {
                                                                          for(l=0;l<z;l++)if(r==h-
>b[y][l])flag1=1;

        if((rel_dt[r]<=y)&&(flag1==0))

                                                                          {
                                                                                  h->b[y][z]=r;
                                                                                  flag=1;
                                                                          }
                                                                  }

                                                          }
                                                  }
                                          }


                //repairing of the chromosome for continuity constraint
                        for(xx=0;xx<NUM_SCR-2;xx++)
                            for(y=2;y<NUM_WEEKS;y++)
                                    for(z=0;z<NUM_SCR;z++)
                                            {  flag1=1;
                                                    r=h->b[y][z];
                                                    flag=1;
                                                    for(k=0;k<NUM_SCR;k++)
                                                            if(r==h->b[y-1][k]){flag=0;flag1=0;}
                                                    if(flag==1)
                                                    {  for(m=0;m<y-1;m++)
                                                            for(k=0;k<NUM_SCR;k++)
                                                                    if(r==h-
>b[m][k]){flag=0;break;}
                                                    }

                                                    if((flag==0)&&(flag1==1))
                                                            {     count=0;
                                                                    for(m=0;m<y-1;m++)
                                                    for(k=0;k<NUM_SCR;k++)
                                                    if(r==h->b[m][k]){count++;}
                                                    k=0;
                                                    flag2=0;
                                                    if(count>NUM_WEEKS/2)
                                    {
        while((flag2==0)&&(k<NUM_SCR))
                                                          {
                                                                flag1=0;
                                                                n=h->b[y-1][k];

        for(l=0;l<NUM_SCR;l++)
                                                                if(n==h-
>b[y][l]){flag1=1;}
        if((flag1==0)&&(commit[y-1][k]==99))
                                                                      {

        h->b[y-1][k]=r;

        flag2=1;
                                                                      }

                                                                k=k+1;
                                                          }
                                    }
                                                    else
 if((count<=NUM_WEEKS/2)&&(commit[y][z]==99)&&(k<NUM_SCR))
                                    {
                                                          while(flag2==0)
                                                          {
                                                                flag1=0;
                                                                n=h->b[y-1][k];

        for(l=0;l<NUM_SCR;l++)

        if(n==h->b[y][l])flag1=1;
                                                                if(flag1==0)
                                                                      {

        h->b[y][z]=n;

        flag2=1;
                                                                      }
                                                                k++;
                                                          }
                                    }
                                                 }
                                    }
                          //checking whether same movie has been assigned on diffenent screens
  in a given week
                                    flag7=1;
                                    for(y=0;y<NUM_WEEKS;y++)
        {
                for(z=0;z<NUM_SCR;z++)
                {
                        for(l=0;l<z;l++)
                        {
                                if(h->b[y][z]==h->b[y][l])
                                {printf("\nThe Two movies are
same on the differenr screen in generation loopbefore obligation cons and no is  %d\n\n",x);
                                flag7=0;break;}
                        }

                }
        }
        if(flag7==0)
        {
                printf("\nthe loop in which the movies are
same\n");
                for(y=0;y<NUM_WEEKS;y++)
                {
                        for(z=0;z<NUM_SCR;z++)
                        printf("%d ",h->b[y][z]);
                        printf("\n");
                }

        }



        //making chromosome feasible for obligation period
constratint
        for(y=0;y<NUM_WEEKS;y++)
        for(z=0;z<NUM_SCR;z++)
        {
                if(y!=NUM_WEEKS-1)
                {
                        flag=0;
                        k=h->b[y][z];
                        if(y!=0)
                        {
                                for(i=0;i<y;i++)
                                for(j=0;j<NUM_SCR;j++)
                                {
                                if(k==h->b[i][j]){
                                flag=1;break;}
                                }
                        }
                        if(flag==0)
                        {
                                rlength=0;
                                for(i=y;i<NUM_WEEKS;i++)
                                for(j=0;j<NUM_SCR;j++){
                                if(k==h->b[i][j]){rlength++;}}
                                if(rlength<OPD[k])
                                {
```

```
                for(l=0;l<NUM_SCR;l++)
                                                                   {
                flag2=1;flag4=1;
                                                                         t=h->b[y+1][l];
                for(i=0;i<=y;i++)
                for(j=0;j<NUM_SCR;j++){
                                                                         if(t==h-
          >b[i][j]){flag4=0;break;}}
                                                 if((commit[y+1][l]==99)&&flag4)
                                                                         {
                                                                               h-
          >b[y+1][l]=k;
                flag2=0;break;
                                                                         }
                                                                   }
                                                             }
                                                       }
                                                }
                                  else
                                  {
                                      flag=0;
                                                       k=h->b[y][z];
                                      for(i=0;i<NUM_WEEKS-1;i++)
                                                       for(j=0;j<NUM_SCR;j++)
                                                       {
                                                             if(k==h->b[i][j]){
                                                             flag=1;break;}
                                                       }
                                      if(flag==0)
                                      {
                                          for(l=0;l<NUM_SCR;l++)
                                          {
                                              flag1=1;
                                                                    t=h->b[y-1][l];
                                              for(m=0;m<NUM_SCR;m++)
                                              if(t==h->b[y][m]){flag1=0;break;}
                                              if(flag1==1){h->b[y][z]=t;break;}
                                          }
                                      }
                                  }
                                          }
                              flag8=1;
```

```
                                        for(y=0;y<NUM_WEEKS;y++)
                                        {
                                              for(z=0;z<NUM_SCR;z++)
                                              {
                                                    for(l=0;l<z;l++)
                                                    {
                                                          if(h->b[y][z]==h->b[y][l])
                                                          {printf("\nThe Two movies are
same on the differenr screen in generation loop and the no is %d\n\n",x);
                                                          flag8=0;break;}
                                                    }

                                              }
                                        }
                                        if(flag8==0)
                                        {
                                              printf("\nthe loop in which the movies are
same\n");

                                              for(y=0;y<NUM_WEEKS;y++)
                                              {
                                                    for(z=0;z<NUM_SCR;z++)
                                                    printf("%d ",h->b[y][z]);
                                                    printf("\n");
                                              }

                                        }

                            h->revenue=evaluation(h->b);
                  if(x!=(popsize-1))
                            {
                                  h->next=(struct chrom *)malloc(sizeof(struct chrom));
                                  if(h->next==NULL)
                                  {
                                        printf("could not assign2");exit(1);
                                  }
                            }
                  else{h->next=NULL;}
                            h=h->next;
                            x++;
                   }while(h!=NULL);
              h=head;
              if(h==NULL){printf("\nERROR1");}
          //finding the best chromosome out of the generated chromosome
              best=((struct chrom *)malloc(sizeof(struct chrom)));
          if(best==NULL){printf("\ncouldnot assign3");}
              best->revenue=h->revenue;
              for(y=0;y<NUM_WEEKS;y++)
              for(z=0;z<NUM_SCR;z++)
              best->b[y][z]=h->b[y][z];
              for(x=0;x<popsize;x++)
              {
                    if(h->revenue>best->revenue)
                    {
                          best->revenue=h->revenue;
                          for(y=0;y<NUM_WEEKS;y++)
                    for(z=0;z<NUM_SCR;z++)
                    best->b[y][z]=h->b[y][z];

          }
          h=h->next;
  }
  printf("\nbest revenue = %f",best->revenue);//getch();
  for(y=0;y<NUM_WEEKS;y++)
  {
          for(z=0;z<NUM_SCR;z++)
           printf(" %d  ",best->b[y][z]);
           printf("\n");

  }


  h=head;

 printf("Please wait till I display the result");

 for(k=0;k<maxgen;k++)
  {
           srand(rrr);
           selection();
           crossover();
           srand(rrr);
           mutation();
           h=head;
           if(head)
           p=head;
           else
           printf("head is null\n");
           q=head;
     for(x=0;x<popsize;x++)
           {
                   if(h && p && h->revenue > p->revenue )
                   {
                           p= h;
                   }
                   if(h && q && h->revenue<q->revenue)
                   {
                           q=h;
                   }
                   h = h->next;
           }
           if(best->revenue>q->revenue)
           {
                   q->revenue=best->revenue;
                   for(y=0;y<NUM_WEEKS;y++)
                   for(z=0;z<NUM_SCR;z++)
                   q->b[y][z]=best->b[y][z];
      }
           if(best->revenue>p->revenue)
           {
                   p->revenue=best->revenue;
```

```
                    for(y=0;y<NUM_WEEKS;y++)
                    for(z=0;z<NUM_SCR;z++)
                    p->b[y][z]=best->b[y][z];
                }
                else
                {
                     best->revenue=p->revenue;
                     for(y=0;y<NUM_WEEKS;y++)
                     for(z=0;z<NUM_SCR;z++)
                     best->b[y][z]=p->b[y][z];
                }


                if(((k+1)%100==0)||(k==maxgen-1))
              {
                     printf("\n%d    %f    %f",popsize, pc,pm);
                     printf("best revenue = %f , generation %d",best->revenue,k+1);
                }
          //In case of applying the screen allotement heuristic (For comparison with
SilverScreener approach)
           /*if(k==(maxgen-1))
              {
                     printf("\n THE MOVIE SHEDULE IS AS FOLLOWS\n\n");
              printf("SCREEN/WEEKS  ");
              //Arranging the movies in descinding order of their demand
                     for(i=0;i<NUM_WEEKS;i++)
              for(j=0;j<=NUM_SCR;j++)
              {

                   for(k=j;k<=NUM_SCR-1;k++)
                   {
                      num1=best->b[i][j];
                      num2=best->b[i][k];
                                 if(demand[num2][i]>demand[num1][i])
                      {
                         temp1=best->b[i][k];
                         best->b[i][k]=best->b[i][j];
                         best->b[i][j]=temp1;
                      }
                   }

               }
                //Movies arranging in descending order with the commitment constraint
                for(n=0;n<2;n++)
                {
                        for(i=0;i<NUM_WEEKS;i++)
                {
                            for(j=0;j<NUM_SCR;j++)
                  {
                  if((commit[i][index_cap[j]]!=99)&&(best->b[i][j]!=commit[i][index_cap[j]]))
                  {
                     for(t=0;t<NUM_SCR;t++){
                     if(best->b[i][t]==(commit[i][index_cap[j]]))break;}
                     temp1=best->b[i][j];
                     best->b[i][j]=best->b[i][t];
                     if(t>j)
                {
                    for(l=t;l>j+1;l--)
                    best->b[i][l]=best->b[i][l-1];
                    best->b[i][j+1]=temp1;
                }
                else
                {
                    for(l=t;l<j-1;l++)
                    best->b[i][l]=best->b[i][l+1];
                    best->b[i][j-1]=temp1;
                }
            }
            }
        }

       }
               for(i=0;i<NUM_SCR;i++)
       printf("%d          ",index_cap[i]+1);
       printf("\n\n");
               for(y=0;y<NUM_WEEKS;y++)
       {
                       printf("week %d          ",y+1);
                       for(z=0;z<NUM_SCR;z++)
           {
                        j=best->b[y][z];
                               printf("%d       ", best->b[y][z]);
           }
                       printf("\n");
               }
       printf("\n\n THE MOVIE SHEDULE IS AS FOLLOWS\n\n");
       printf("WEEK/SCREEN    ");
       for(i=0;i<NUM_WEEKS;i++)
       printf("%-10d ",i+1);
       printf("\n\n");
               for(z=0;z<NUM_SCR;z++)
       {
                       printf("screen %d ",index_cap[z]+1);
                       for(y=0;y<NUM_WEEKS;y++)
           {
                        j=best->b[y][z];
                               printf("%-20s    ", movie[j]);
           }
                       printf("\n");
               }

          }*/

     if(k==(maxgen-1))
       {
               printf("\n THE MOVIE SHEDULE IS AS FOLLOWS\n\n");
        printf("SCREEN/WEEKS          ");
        for(y=0;y<NUM_WEEKS;y++)
        {
                       printf("week %d          ",y+1);
                       for(z=0;z<NUM_SCR;z++)
           {
                         j=best->b[y][z];
                               printf("%d        ", best->b[y][z]);
           }
```

```
                        printf("\n");
                  }
         printf("\n\n THE MOVIE SHEDULE IS AS FOLLOWS\n\n");
         printf("WEEK/SCREEN  ");
         for(i=0;i<NUM_WEEKS;i++)
         printf("%-10d ",i+1);
         printf("\n\n");
                  for(z=0;z<NUM_SCR;z++)
         {
                          printf("screen %d ",index_cap[z]+1);
                          for(y=0;y<NUM_WEEKS;y++)
             {
                            j=best->b[y][z];
                                printf("%-20s    ", movie[j]);
             }
                           printf("\n");
                   }

            }


            gen++;
         if(best->revenue>highest){highest=best->revenue;hpop=popsize;hpc=pc;hpm=pm;}

      }
            best_revenue[ab][bc]+=best->revenue;
             p=head;q=head;
            while(q!=NULL)
                  {
                       p=q->next;
                       free(q);
                       q=p;
                  }
                  free(best);
   }

            tot_pop0=0;tot_pop1=0;tot_pc0=0;tot_pc1=0;tot_pm0=0;tot_pm1=0;
            for(bc=0;bc<2;bc++)
            for(cd=0;cd<2;cd++)
                   {
                           tot_pop0+=best_revenue[0][bc];
                           tot_pop1+=best_revenue[1][bc];
                   }
             for(ab=0;ab<2;ab++)
         for(cd=0;cd<2;cd++)

                   {
                           tot_pc0+=best_revenue[ab][0];
                           tot_pc1+=best_revenue[ab][1];

                   }
         for(ab=0;ab<2;ab++)
         for(bc=0;bc<2;bc++)

                   {          tot_pc0+=best_revenue[ab][bc][0];
```

```
                                    tot_pc1+=best_revenue[ab][bc][1];
                            }
                  if(tot_pop0>tot_pop1)best_pop=300;else best_pop=400;
                  if(tot_pc0>tot_pc1)best_pc=0.95;else best_pc=0.99;
          if(tot_pm0>tot_pm1)best_pm=0.03;else best_pm=0.05;
          printf("\n");
                  printf(" best popsize=%d, bestpc=%f, ",best_pop,best_pc);
                  printf("\nhighest rev=%f,hpop=%d,hpc=%f,hpm=%f",highest, hpop,hpc,hpm);
                  return(0);
}


void selection()
{
int sel[500],flag,sel_ar[500];
int x,y,z,r,n,k;
struct chrom *p,*q;
flag=1;

p=head;

for(n=0;n<2;n++)
{
          z=popsize;
          p=head;
          x=0;
          while(p!=NULL)
          {
                    p->no=x;
                    x++;
                    p=p->next;
          }

          for(x=0;x<popsize;x++)sel[x]=x;
          for(x=0;x<popsize-2;x++)
          {
                    r=rand()% z;
                    sel_ar[x]=sel[r];
                    for(y=r;y<z;y++)sel[y]=sel[y+1];
                    z-=1;
          }
          sel_ar[x]=sel[0];
          sel_ar[x+1]=sel[1];

          for(x=0;x<popsize-1;x++)
          {         y=sel_ar[x];
                    p=head;q=head;
                    while(p->no!=y){p=p->next;}
                    z=sel_ar[x+1];
                    while(q->no!=z){q=q->next;}
                    if(p->revenue>q->revenue)p->copies++;else q->copies++;
                    x++;
           }
    }
```

```
while(flag)
{

        if(head->copies==0)
    {

        p=head->next;
          free(head);

          head=p;
          flag=1;

    }

    else

    flag=0;

}

p=head;x=0;

if(p==NULL){printf("ERROR\n");}

q=head->next;
while(q!=NULL)

{

        if(q->copies==0)
    {

        p->next=q->next;

          free(q);

          q=p->next;

    }

    else

    {

        p=q;
          q=p->next;

    }

}

p=head;

//making copies of chromosome
```

```
while(p!=NULL)
{
        z=p->copies;
        if(z!=0)
        for(x=2;x<=z;x++)
        {
                q=(struct chrom *)(malloc(sizeof(struct chrom)));
          if(q==NULL){printf("couldn't assign3\n");}

                q->revenue=p->revenue;
                q->copies=0;
                for(y=0;y<NUM_WEEKS;y++)
                for(k=0;k<NUM_SCR;k++)
                {q->b[y][k]=p->b[y][k];}
                q->next=p->next;
                p->next=q;
        }
        p=p->next;
 }
 p=head; x=0;
 while(p!=NULL)
 {
        p->no=x;
     if(x==(popsize-1)){p->next=NULL;}
        p->copies=0;
        p=p->next;
        x++;
 }
p=head;

}

//In case of applying the screen allotement heuristic (For comparison with SilverScreener
approach)
/*float evaluation(int b[NUM_WEEKS][NUM_SCR])
{
 int x=0,y=0,k=0;
 float revenue=0.0,temp;
 for(x=0;x<NUM_WEEKS;x++)
        for(y=0;y<NUM_SCR;y++)
        {
         k=b[x][y];

                revenue=revenue+(0.133* demand[k][x]+0.9434* demand[k][x]*(1-
(contract_term[k][x])/100.0));
        }
 return(revenue);
 } */


 float evaluation(int b[NUM_WEEKS][NUM_SCR])
 {
  int x=0,y=0,k=0;
  float revenue=0.0,temp;
  for(x=0;x<NUM_WEEKS;x++)
```

```
        for(y=0;y<NUM_SCR;y++)
        {
          k=b[x][y];
                if((demand[k][x] < screen_cap[y])&&(demand[k][x]>=0))
                temp= demand[k][x];
                else
                temp=screen_cap[y];

                revenue=revenue+(0.133*temp+0.9434*temp*(1-(contract_term[k][x])/100.0));
        }
    return(revenue);
    }


    void crossover()
    {
            int x,y,i,j,xx,z,c,k,r,r1,r2,l,temp,count,e,flag3,flag4;
            struct chrom *p,*q;
            int cross[1500],flag,flag1,flag2,m,n;
            int crosspair[1500];
            z=popsize;
            p=head;if(p==NULL){printf("ERROR3\n");}
            x=0;
            while(p!=NULL)
                    {
                    p->no = x;
                    x++;
                    p=p->next;
                    }
            for(x=0;x<popsize;x++)cross[x]=x;
            for(x=0;x<popsize-2;x++)
                    {
                            r=rand()%z;
                            crosspair[x]=cross[r];
                            for(y=r;y<z;y++)cross[y]=cross[y+1];
                            z-=1;
                    }
            crosspair[x]=cross[0];
            crosspair[x+1]=cross[1];
            for(x=0;x<(popsize-1);x++)
                    {       y=crosspair[x];
                            p=q=head;
                            for(p=head;p && p->no!=y;p=p->next);
                            e=crosspair[x+1];
                            for(q=head;q && q->no!=e;q=q->next);
                            r= rand()% 101;
                            if(r<pc)
                                    {
                                            r1=rand()% MAXSLOT;
                                            r2=(r1%NUM_SCR);

                                                            for(k=r2;k<NUM_WEEKS;k++)
                                                            for(l=0;l<NUM_SCR;l++)
                                                                    {
                                                                            temp=p->b[k][l];
```

```
                                        p->b[k][l]=q->b[k][l];
                                        q->b[k][l]=temp;
                                                    }

                                }


                    }
        p=head;if(p==NULL){printf("ERROR4\n");}
        c=0;
        while(p!=NULL)
        {

                for(xx=0;xx<NUM_SCR-2;xx++)
                        for(y=2;y<NUM_WEEKS;y++)
                                for(z=0;z<NUM_SCR;z++)
                                        {  flag1=1;
                                            r=p->b[y][z];
                                            flag=1;
                                            for(k=0;k<NUM_SCR;k++)
                                                    if(r==p->b[y-1][k]){flag=0;flag1=0;}
                                            if(flag==1)
                                            {  for(m=0;m<y-1;m++)
                                                    for(k=0;k<NUM_SCR;k++)
                                                            if(r==p-
    >b[m][k]){flag=0;break;}
                                            }

                                            if((flag==0)&&(flag1==1))
                                                    {     count=0;
                                                            for(m=0;m<y-1;m++)
                                                            for(k=0;k<NUM_SCR;k++)
                                                            if(r==p->b[m][k]){count++;}
                                                            k=0;
                                                            flag2=0;
                                                            if(count>NUM_WEEKS/2)
                                        {

        while((flag2==0)&&(k<NUM_SCR))
                                                                    {
                                                                            flag1=0;
                                                                            n=p->b[y-1][k];


        for(l=0;l<NUM_SCR;l++)

        if(n==p->b[y][l])flag1=1;

        if((flag1==0)&&(commit[y-1][k]==99))
                                                                                {

        p->b[y-1][k]=p->b[y][z];

        flag2=1;
                                                                                }
```

```
                                                                      k++;
                                                                }
                                      }
                                                         else
if((count<=NUM_WEEKS/2)&&(commit[y][z]==99)&&(k<NUM_SCR))
                                                         while(flag2==0)
                                                                {
                                                                      flag1=0;
                                                                      n=p->b[y-1][k];


      for(l=0;l<NUM_SCR;l++)

      if(n==p->b[y][l])flag1=1;
                                                                      if(flag1==0)
                                                                            {


      p->b[y][z]=n;

      flag2=1;
                                                                            }
                                                                      k++;
                                                                }
                                                    }
                                       }


            p=p->next;c++;
      }
      p=head;
      if(p==NULL)printf("error4\n");
      c=0;
      while(p!=NULL)
      {

        for(y=0;y<NUM_WEEKS;y++)
             for(z=0;z<NUM_SCR;z++)
             {
                   if(y!=NUM_WEEKS-1)
                   {
                         flag=0;
                         k=p->b[y][z];
                         if(y!=0)
                         {
                               for(i=0;i<y;i++)
                               for(j=0;j<NUM_SCR;j++)
                               {
                                     if(k==p->b[i][j]){
                                     flag=1;break;}
                               }
                         }
                         if(flag==0)
                         {
                               rlength=0;
```

```
                                        for(i=y;i<NUM_WEEKS;i++)
                                        for(j=0;j<NUM_SCR;j++){
                                        if(k==p->b[i][j]){rlength++;}}
                                        if(rlength<OPD[k])
                                        {
                                              for(l=0;l<NUM_SCR;l++)
                                              {
                                                    flag2=1;flag4=1;
                                                    t=p->b[y+1][l];

                                                          for(i=0;i<=y;i++)
                                                          for(j=0;j<NUM_SCR;j++){
                                                          if(t==p-
>b[i][j]){flag4=0;break;}}

                                                    if((commit[y+1][l]==99)&&flag4)
                                                    {
                                                          p->b[y+1][l]=k;
                                                          flag2=0;break;
                                                    }
                                              }
                                        }
                                  }
                            }
              else
              {
               flag=0;
                          k=p->b[y][z];
                 for(i=0;i<NUM_WEEKS-1;i++)
                          for(j=0;j<NUM_SCR;j++)
                          {
                                if(k==p->b[i][j]){
                                flag=1;break;}
                 }
                 if(flag==0)
                 {
                    for(l=0;l<NUM_SCR;l++)
                    {
                          flag1=1;
                                      t=p->b[y-1][l];
                       for(m=0;m<NUM_SCR;m++)
                       if(t==p->b[y][m]){flag1=0;break;}
                       if(flag1==1){p->b[y][z]=t;break;}
                    }
                 }
              }


               }


              p=p->next;c++;
               }
```

```
        p=head;
        while(p!=NULL)
                {
                        p->revenue=evaluation(p->b);
                        p=p->next;
                }
}

void mutation()
{
        struct chrom *p;
        int x,c,y,z,k,r,m,n,l,flag,flag1,flag2,flag3,flag4,xx,yy,ch_mov,rr,temp1,xxx,count;
        flag4=0;temp1=0;
     c=0;
     p=head;
        while(p!=NULL)
        {
                for(xx=0;xx<NUM_WEEKS;xx++)
                {
                for(yy=0;yy<NUM_SCR;yy++)
                {
                r=rand()%101;

                if((r<pm)&&(commit[xx][yy]==99))
                        {
                                flag1=1;

                                for(x=0;x<NUM_MOVIES;x++)
                                {       flag=0;
                        for(y=0;y<NUM_WEEKS;y++)
                                        for(z=0;z<NUM_SCR;z++)
                                                {
                                                        if(x==p->b[y][z])
                                                                {
                                                                        flag=1;
                                                                }
                                                }

if((flag==0)&&(rel_dt[x]<=xx)){ch_mov=x;flag1=0;break;}
                                }
                   if(flag1==0)p->b[xx][yy]=ch_mov;
                   else
                                {
                                        for(x=0;x<NUM_MOVIES;x++)
                                        {       flag=0;
                                                for(z=0;z<NUM_SCR;z++)
                                                        {
                                                                if(x==p->b[xx][z])
                                                                        {
                                                                                flag=1;
                                                                        }
                                                        }
                                                if((flag==0)&&(rel_dt[x]<xx)){p->b[xx][yy]=x;
 break;}
                                        }
```

```
                                            }


                                                if((r<pm/2)&&(commit[xx][yy]==99))
                                                {
                                                    do
                                                    {
                                                            rr=rand()%NUM_SCR;
                                                    }while(commit[xx][rr]!=99);
                                                    temp1= p-
>b[xx][rr];if((temp1>NUM_MOVIES)||(temp1<0)){printf("oh no %d",temp1);/*getch();*/}
                                                    p->b[xx][rr]=p->b[xx][yy]; if((p-
>b[xx][rr]>NUM_MOVIES)||(p->b[xx][rr]<0)){printf("oh no1 %d",p->b[xx][rr]);/*getch();*/}
                                                    p->b[xx][yy]=temp1;if((p-
>b[xx][yy]>NUM_MOVIES)||(p->b[xx][yy]<0)){printf("oh no2");/*getch();*/}
                                                }
                                                if((gen==maxgen-1))
 {/*printf("\n")*/;for(y=0;y<NUM_WEEKS;y++)
                                    {for(z=0;z<NUM_SCR;z++)
                                    if((p->b[y][z]>NUM_MOVIES)||(p->b[y][z]<0)){
                                    printf("\ni am here%d no. %d",p->b[y][z],p->no);}}}

                                    }
                            }
                            }
                            for(xxx=0;xxx<NUM_SCR-2;xxx++)
                            for(y=2;y<NUM_WEEKS;y++)
                                            for(z=0;z<NUM_SCR;z++)
                                                    {  flag1=1;
                                                            r=p->b[y][z];
                                                            flag=1;
                                                            for(k=0;k<NUM_SCR;k++)
                                                                    if(r==p->b[y-1][k]){flag=0;flag1=0;}
                                                            if(flag==1)
                                                            {  for(m=0;m<y-1;m++)
                                                                    for(k=0;k<NUM_SCR;k++)
                                                                            if(r==p-
 >b[m][k]){flag=0;break;}
                                                            }

                                                            if((flag==0)&&(flag1==1))
                                                                    {     count=0;
                                                                            for(m=0;m<y-1;m++)
                                                                            for(k=0;k<NUM_SCR;k++)
                                                                            if(r==p->b[m][k]){count++;}
                                                                            k=0;
                                                                            flag2=0;
                                                                            if(count>NUM_WEEKS/2)
                                                    {

                    while((flag2==0)&&(k<NUM_SCR))
                                                                                    {
                                                                                            flag1=0;
                                                                                            n=p->b[y-1][k];
                                                for(l=0;l<NUM_SCR;l++)
>b[y][l])flag1=1;                                                         if(n==p-

        if((flag1==0)&&(commit[y-1][k]==99))
                                                                                {

        p->b[y-1][k]=p->b[y][z];

        flag2=1;

                                                                                }

                                                                        k++;
                                                                  }
                                        }
                                                              else
if((count<NUM_WEEKS/2)&&(commit[y][z]==99)&&(k<NUM_SCR))
                                        {
                                                                  while(flag2==0)
                                                                  {
                                                                        flag1=0;
                                                                        n=p->b[y-1][k];


        for(l=0;l<NUM_SCR;l++)

        if(n==p->b[y][l])flag1=1;
                                                                        if(flag1==0)
                                                                              {

        p->b[y][z]=n;

        flag2=1;

                                                                              }
                                                                        k++;
                                                                  }
                                        }
                                                    }
                                         }

              if((gen==maxgen-1)) {for(y=0;y<NUM_WEEKS;y++)
              {for(z=0;z<NUM_SCR;z++)
              if((p->b[y][z]>NUM_MOVIES)||(p->b[y][z]<0)){
              printf("\n%d no. %d",p->b[y][z],p->no);}}}


                 p=p->next;  c++;
        }


        p=head;  c=0;
        while(p!=NULL)
        {
          for(y=0;y<NUM_WEEKS;y++)
              for(z=0;z<NUM_SCR;z++)
```

```
                {
                        if(y!=NUM_WEEKS-1)
                        {
                                flag=0;
                                k=p->b[y][z];
                                if(y!=0)
                                {
                                        for(i=0;i<y;i++)
                                        for(j=0;j<NUM_SCR;j++)
                                        {
                                                if(k==p->b[i][j]){
                                                flag=1;break;}
                                        }
                                }
                                if(flag==0)
                                {
                                        rlength=0;
                                        for(i=y;i<NUM_WEEKS;i++)
                                        for(j=0;j<NUM_SCR;j++){
                                        if(k==p->b[i][j]){rlength++;}}
                                        if(rlength<OPD[k])
                                        {
                                                for(l=0;l<NUM_SCR;l++)
                                                {
                                                        flag2=1;flag4=1;
                                                        t=p->b[y+1][l];

                                                                for(i=0;i<y;i++)
                                                                for(j=0;j<NUM_SCR;j++){
                                                                if(t==p-
>b[i][j]){flag4=0;break;}}

                                                        if((commit[y+1][l]==99)&&flag4)
                                                        {
                                                                p->b[y+1][l]=k;
                                                                flag2=0;break;
                                                        }
                                                }
                                        }
                                }
                        }
                else
                {
                 flag=0;
                                k=p->b[y][z];
                    for(i=0;i<NUM_WEEKS-1;i++)
                                for(j=0;j<NUM_SCR;j++)
                                {
                                        if(k==p->b[i][j]){
                                        flag=1;break;}
                    }
                    if(flag==0)
                    {
                       for(l=0;l<NUM_SCR;l++)
                        {
                                flag1=1;
                                            t=p->b[y-1][l];
                              for(m=0;m<NUM_SCR;m++)
                              if(t==p->b[y][m]){flag1=0;break;}
                              if(flag1==1){p->b[y][z]=t;break;}
                          }
                      }
                  }


                  }


                          c++;
                          p=p->next;
            }




            p=head;
            while(p!=NULL)
                  {
                          p->revenue=evaluation(p->b);
                          p=p->next;
                  }

      }


void readdata()
{
            FILE *fp1;
        int min=0;
        int i,j,rel[NUM_MOVIES];
            fp1=fopen("mt_35-39.txt","r+");
            if(fp1==NULL){printf("error"); }
            fscanf(fp1,"%*s %*s ");
            //printf("the obligation periods for movies \n");
            for(i=0;i<NUM_MOVIES;i++)
            {
              fscanf(fp1,"%*d %d",&OPD[i]);
                    printf("%d ",OPD[i]);
            }

            fscanf(fp1,"%*s %*s");
            printf("the release dates for movies \n");
            for(i=0;i<NUM_MOVIES;i++)
            {
                    fscanf(fp1,"%*d %d %s",&rel[i],movie[i]);
              if(i==0)min=rel[i];
              else if (min>rel[i]){min=rel[i];}
        }
        printf("the min is %d",min);
```

```
    for(i=0;i<NUM_MOVIES;i++)
    rel_dt[i]=rel[i]-min;
    printf("\nthe movies name are as follows\n");
    for(i=0;i<NUM_MOVIES;i++)
    printf("\n %s",movie[i]);
        printf("the revenue forecasts for movies \n");
        fscanf(fp1,"%*s %*s ");
        for(i=0;i<NUM_WEEKS;i++)
        fscanf(fp1," %*s");
        for(i=0;i<NUM_MOVIES;i++)
        {
                fscanf(fp1," %*d");
                for(j=0;j<NUM_WEEKS;j++)
                {
                        fscanf(fp1," %f",&demand[i][j]);
        }
                fscanf(fp1," %*s");
    }

    //printf("\nthe exhibitor share for movies \n");
        fscanf(fp1,"%*s %*s ");
        for(i=0;i<NUM_WEEKS;i++)
        fscanf(fp1," %*s");
        for(i=0;i<NUM_MOVIES;i++)
        {
                fscanf(fp1," %*d");
                for(j=0;j<NUM_WEEKS;j++)
                {
                        fscanf(fp1," %f",&contract_term[i][j]);
                        printf("%3.2f ",contract_term[i][j]);
                }
                fscanf(fp1," %*s");
                printf("\n");
        }

}
```

INPUT FILE FOR MACRO SCHEDULING (For MUNT Week 35-39)

```
param opd:=
1    2
2    2
3    2
4    2
5    2
6    2
7    2
8    2
9    2
10   2
11   2
12   2
13   2
14   2
15   2
16   2
17   2
18   2
19   2
20   2
21   2
22   2
23   2
24   2
25   2
26   2
27   2

param r:=
1      36     #COMMITTED_DWN
2      36     #CB
3      37     #RR
4      35     #BLW
5      35     #ANML
6      38     #MRG
7      35     #CONTD_BJD2
8      35     #TH
9      35     #HB
10     35     #PH_CMBND
11     35     #SHOV
12     35     #LCTR
13     35     #DD2
14     35     #JPIII
15     35     #SMII-1
16     35     #SMII-2
17     35     #D2E
18     35     #FF_1
19     35     #FF_2
20     35     #TF
21     35     #FORTHCOMING_POA
22     35     #SA
23     36     #NNK
```

```
24      36      #TC
25      37      #SWF
26      38      #TFF
27      39      #CCM


param revenue:
        35      36      37      38      39:=
1       -99999  20000   15000   10000   7211    #COMMITTED_DWN
2       -9999   17500   15000   10000   7883    #CB
3       -99999  -99999  55000   45000   30000   #RR
4       28000   18000   10000   6123    3659    #BLW
5       30000   20000   15000   10491   7443    #ANML
6       -99999  -99999  -99999  50000   40000   #MRG
7       23538   20999   19835   18735   17696   #CONTD_BJD2
8       9910    8328    6998    5881    4942    #TH
9       5170    2808    1525    829     450     #HB
10      12485   8497    7010    5783    4771    #PH_CMBND
11      13823   9268    7589    6214    5088    #SHOV
12      9785    7217    5323    3926    2896    #LCTR
13      6036    3475    2636    2000    1517    #DD2
14      10702   3809    2272    1356    809     #JPIII
15      1601    928     538     312     181     #SMII-1
16      7084    4108    2383    1382    801     #SMII-2
17      19307   11584   8973    6951    5384    #D2E
18      5196    4025    3117    2415    1870    #FF_1
19      27754   16652   12899   9991    7739    #FF_2
20      9058    8745    8443    8152    7870    #TF
21      80000   60000   46978   36389   28187   #FORTHCOMING_POA
22      15000   12000   9196    7123    5517    #SA
23      -99999  30000   20000   20000   15263   #NNK
24      -99999  25000   20000   12000   8722    #TC
25      -9999   -9999   40000   32000   28000   #SWF
26      -99999  -99999  -99999  75000   40000   #TFF
27      -99999  -99999  -99999  -99999  35000   #CCM

param share:

        35      36      37      38      39:=
1       0       38      38      38      35.5    #COMMITTED_DWN
2       0       38      38      38      35.5    #CB
3       0       0       50      45      40      #RR
4       50      45      40      37.5    35      #BLW
5       50      45      40      37.5    35      #ANML
6       0       0       0       50      50      #MRG
7       27.5    27.5    27.5    27.5    27.5    #CONTD_BJD2
8       30      27.5    27.5    27.5    27.5    #TH
9       32.5    30      27.5    27.5    27.5    #HB
10      27.50   27.50   27.50   27.50   27.50   #PH_CMBND
11      30      27.5    27.5    27.5    27.5    #SHOV
12      30      27.5    27.5    27.5    27.5    #LCTR
13      30      27.5    27.5    27.5    27.5    #DD2
14      37.5    35      32.5    30      27.5    #JPIII
15      35.5    33      30.5    28      27.5    #SMII-1
```

16	35.5	33	30.5	28	27.5	
17	35	32.5	30	27.5	27.5	#SMII-2
18	40	37.5	35	32.5	27.5	#D2E
19	40	37.5	35	32.5	30	#FF_1
20	27.5	27.5	27.5	27.5	30	#FF_2
21	60	60	57.5	55	27.5	#TF
22	45	40	37.5	35	52.5	#POA
23	0	50	50	40	32.5	#SA
24	0	50	40	35	37.5	#NNK
25	0	0	50	50	32.5	#TC
26	0	0	0	50	40	#SWF
27	0	0	0	0	45	#TFF
					50	#CCM

APPENDIX-1 (C)
Macro Schedules Generated by SilverScreener Implementation Approach and GA Based Approach

The Movie Schedule by GA Based Approach for Theatre 1 (Week 36-39) is as Follows

	14	13	4	11	3	5	10	12	2	9	1	8	6	7
Week 1	17	19	21	3	18	16	4	20	5	12	11	8	7	13
Week 2	17	1	2	19	18	4	16	3	5	21	12	11	8	7
Week 3	15	2	1	0	17	19	18	16	4	5	8	12	7	3
Week 4	15	2	1	0	19	17	18	16	5	4	8	12	7	3

THE MOVIE SHEDULE IS AS FOLLOWS

WEEK/SCREEN	1	2	3	4
screen 14	#POA	#POA	#TFF	#TFF
screen 13	#BTC	#RR	#SWF	#SWF
screen 4	#DRI	#SWF	#RR	#RR
screen 11	#DWN	#BTC	#COMMITTED_MRG	#COMMITTED_MRG
screen 3	#ANML	#ANML	#POA	#BTC
screen 5	#SA	#CB	#BTC	#POA
screen 10	#CB	#SA	#ANML	#ANML
screen 12	#BLW	#DWN	#SA	#SA
screen 2	#CONTD_BJD	#CONTD_BJD	#CB	#CONTD_BJD
screen 9	#FF	#DRI	#CONTD_BJD	#CB
screen 1	#SMII	#FF	#SHO	#SHO
screen 8	#SHO	#SMII	#FF	#FF
screen 6	#PH	#SHO	#PH	#PH
screen 7	#D2E	#PH	#DWN	#DWN

The Movie Schedule Recommended by SilverScreener Approach for Theatre 1 (Week 36-39) is as Follows

screen 14	602	PLANET OF THE APES	PLANET OF THE APES	FAST & THE FURIOUS	E FAST & THE FURIOUS
screen 13	322	BLESS THE CHILD	RAT RACE	SWORDFISH	SWORDFISH
screen 4	282	ANIMAL	SWORDFISH	RAT RACE	AIN CORRELLI'S MANDOLIN
screen 11	282	DOWN	BLESS THE CHILD	MOULIN ROUGE	MOULIN ROUGE
screen 3	205	DRIVEN	ANIMAL	ANET OF THE APES	RAT RACE
screen 5	205	SOUL ASSASSIN	CRAZY/BEAUTIFUL	BLESS THE CHILD	BLESS THE CHILD
screen 10	205	CRAZY/BEAUTIFUL	SOUL ASSASSIN	ANIMAL	PLANET OF THE APES
screen 12	205	BLOW	DOWN	SOUL ASSASSIN	ANIMAL
screen 2	183	DGET JONES DIARY	DGET JONES DIARY	CRAZY/BEAUTIFUL	SOUL ASSASSIN
screen 9	183	FINAL FANTA	DRIVEN	DGET JONES DIARY	RIDGET JONES DIARY
screen 1	161	SCARY MOVIE II	FINAL FANTA	SHREK OV	CRAZY/BEAUTIFUL
screen 8	149	SHREK OV	SCARY MOVIE II	FINAL FANTA	SHREK OV
screen 6	148	PEARL HARBOR	SHREK OV	SCARY MOVIE II	FINAL FANTA
screen 7	118	DOWN TO EARTH	PEARL HARBOR	DOWN	SCARY MOVIE II

The Movie Schedule bv GA Based Approach for Theatre 1 (Week 38-39) is as Follows

	14	13	4	11	3	5	10	12	2	9	1	8	6	7
Week 1	3	21	20	2	17	4	10	6	19	15	16	12	11	7
Week 2	3	4	21	20	1	0	2	17	10	6	15	16	12	5

THE MOVIE SHEDULE IS AS FOLLOWS

WEEK/SCREEN	1	2
screen 14	#TFF	#TFF
screen 13	#SWF	#TFF2
screen 4	#RR	#SWF
screen 11	#MRG	#RR
screen 3	#BLW	#CCM
screen 5	#TFF2	#COMMITTED_DG
screen 10	#SMII	#MRG
screen 12	#CONTD_BJD	#BLW
screen 2	#DWN	#SMII
screen 9	#DRI	#CONTD_BJD
screen 1	#ANML	#DRI
screen 8	#POA	#ANML
screen 6	#D2E	#POA
screen 7	#PH	#SHO

<u>The Movie Schedule Recommended by SilverScreener Approach for Theatre 1 (Week 38-39) is as Follows</u>

Week/Screen	Capacity	1	2
screen 14	602	**The Fast & The Furious**	The Fast & The Furious
screen 13	322	Swordfish	The Fast & The Furious(2)
screen 4	282	Rat Race	Swordfish
screen 11	282	**Moulin Rouge**	Moulin Rouge
screen 3	205	Blow	**Captain Correli's Mandolin**
screen 5	205	**The Fast & The Furious(2)**	**De Grot**
screen 10	205	Scary Movie II	Rat Race
screen 12	205	Bridget Jones Diary	Blow
screen 2	183	Down	Scary Movie II
screen 9	183	Driven	Bridget Jones Diary
screen 1	161	Animal	Driven
screen 8	149	Planet Of The Apes	Animal
screen 6	148	Down To Earth	Planet Of The Apes
screen 7	118	Shrek OV	**Get Carter**

The Movie Schedule by GA Based Approach for Theatre 2 (Week 36-39) is as Follows

THE MOVIE SHEDULE IS AS FOLLOWS

SCREEN/WEEKS	1	7	2	6	5	3	4
Week 1	9	2	11	10	5	7	3
Week 2	0	1	9	2	11	7	5
Week 3	8	0	1	9	7	11	5
Week 4	8	0	1	7	9	11	5

THE MOVIE SHEDULE IS AS FOLLOWS

WEEK/SCREEN	1	2	3	4
screen 1	#BTC	#COMMITTED_RR	#FORTHCOMING_TFF	#FORTHCOMING_TFF
screen 7	=DWN	#SWF	#COMMITTED_RR	#COMMITTED_RR
screen 2	#ANML	#BTC	#SWF	#SWF
screen 6	#DRI	#DWN	#BTC	#SHO
screen 5	=SMII	#ANML	#SHO	#BTC
screen 3	=SHO	#SHO	#ANML	#ANML
screen 4	#CONTD_FF	#SMII	#SMII	#SMII

The Movie Schedule by SilverScreener Approach for Theatre 2 (Week 36-39) is as Follows

Week	1	2	3	4
Screen 1 711	BLESS THE CHILD	RAT RACE	THE FAST & THE FURIOUS	THE FAST & THE FURIOUS
Screen 7 70	DOWN	SWORDFISH	RAT RACE	RAT RACE
Screen 2 200	ANIMAL	BLESS THE CHILD	SWORDFISH	SWORDFISH
Screen 6 124	DRIVEN	DOWN	BLESS THE CHILD	SHREK OV
Screen 5 100	SCARY MOVIE II	ANIMAL	SHREK OV	BLESS THE CHILD
Screen 3 89	SHREK OV	SHREK OV	ANIMAL	ANIMAL
Screen 4 70	FINAL FANTASY	SCARY MOVIE II	SCARY MOVIE II	SCARY MOVIE II

The Movie Schedule by GA Based Approach for Theatre 2 (Week 37-39) is as Follows

THE MOVIE SHEDULE IS AS FOLLOWS

SCREEN/WEEKS	1	7	2	6	5	3	4
Week 1	1	2	12	8	9	3	11
Week 2	11	1	2	8	3	9	12
Week 3	11	1	0	2	3	8	9

THE MOVIE SHEDULE IS AS FOLLOWS

WEEK/SCREEN	1	2	3
screen 1	#RR	#FORTHCOMING_TFF	#FORTHCOMING_TFF
screen 7	#SWF	#RR	#RR
screen 2	#DWN	#SWF	#COMMITTED_DG
screen 6	#ANML	#ANML	#SWF
screen 5	#DRI	#CONTD_SHO	#CONTD_SHO
screen 3	#CONTD_SHO	#DRI	#ANML
screen 4	#FORTHCOMING_TFF	#DWN	#DRI

<u>The Movie Schedule by SilverScreener Approach for Theatre 2 (Week 37-39) is as Follows</u>

Week	1	2	3
Screen 1 711	**Rat Race**	Fast & Furious	Fast & Furious
Screen 7 270	**Swordfish**	Rat Race	Rat Race
Screen 2 200	Down	Swordfish	**De Grot**
Screen 6 124	Animal	Animal	Swordfish
Screen 5 100	Driven	Shrek OV	Shrek OV
Screen 3 89	Shrek OV	Driven	Animal
Screen 4 70	Scary Movie 2	Down	Driven

The Movie Schedule by GA Based Approach for Theatre 3 (Week 37-39)

THE MOVIE SHEDULE IS AS FOLLOWS

SCREEN/WEEKS	11	3	1	2	9	10	8	6	7	4	12	5	13
Week 1	4	1	14	2	5	18	21	19	20	7	9	6	13
Week 2	2	0	4	1	5	14	18	3	7	21	9	19	20
Week 3	0	3	2	1	4	5	14	18	7	21	9	20	19

THE MOVIE SHEDULE IS AS FOLLOWS

WEEK-SCREEN	1	2	3
screen 11	#CONTD_POA	#RR	#COMMITTED_MRG
screen 3	#SWF	#COMMITTED_MRG	#JJ
screen 1	#BLW	#CONTD_POA	#RR
screen 2	#RR	#SWF	#SWF
screen 9	#D2E	#D2E	#CONTD_POA
screen 10	#NNK	#BLW	#D2E
screen 8	#TC	#NNK	#BLW
screen 6	#DWN	#JJ	#NNK
screen 7	#CB	#SMII	#SMII
screen 4	#SMII	#TC	#TC
screen 12	#BJD	#BJD	#BJD
screen 5	=FF	=DWN	#CB
screen 13	#PH_CMBND	#CB	#DWN

The Movie Schedule by SilverScreener Approach for Theatre 3 (Week 37-39)

Week		1	2	3
Scr 11	382	Planet of the Apes	The Fast and the Furious	Moulin Rouge
Scr 3	340	Swordfish	Moulin Rouge	The Fast and the Furious
Scr 1	223	Blow	Rat Race	"Captain Correli's Mandolin
Scr 2	223	Rat Race	Planet of the Apes	Jalla! Jalla!
Scr 9	177	Down to Earth	Swordfish	Rat Race
Scr 10	177	Nynke	Down to Earth	Swordfish
Scr 8	173	The Contender	Blow	Planet of the Apes
Scr 6	163	Down	Jalla! Jalla!	Down to Earth
Scr 7	163	Crazy Beautiful	Nynke	Blow
Scr 4	114	Scary Moyie 2	Scary Movie 2	Nynke
Scr 12	110	Animal	The Contender	Scary Movie 2
Scr 5	104	Bridget Jones Diary	Bridget Jones Diary	The Contender
Scr 13	100	Final Fantasy	"Crazy.'Beautiful"	Bridget Jones Diary

The Movie Schedule by GA Based Approach for Theatre 3 (Week 36-39)

THE MOVIE SHEDULE IS AS FOLLOWS

SCREEN/WEEKS	11	3	1	2	9	10	8	6	7	4	12	5	13
Week 1	6	20	8	3	22	21	9	12	4	5	10	16	13
Week 2	6	1	8	2	20	5	3	4	22	9	21	12	16
Week 3	19	0	2	1	6	20	8	5	22	3	4	12	17
Week 4	0	19	2	1	6	8	20	5	4	22	17	3	12

3 THE MOVIE SHEDULE IS AS FOLLOWS

WEEK/SCREEN	1	2	3	4
screen 11	#CONTD_POA	#CONTD_POA	#TFF	#COMMITTED_MRG
screen 3	#NNK	#SWF	#COMMITTED_MRG	#TFF
screen 1	#D2E	#D2E	#RR	#RR
screen 2	#DWN	#RR	#SWF	#SWF
screen 9	#ANML	#NNK	#CONTD_POA	#CONTD_POA
screen 10	#BLW	#TC	#NNK	#D2E
screen 8	#FF	#DWN	#D2E	#NNK
screen 6	#BJD2	#CB	#TC	#TC
screen 7	#CB	#ANML	#ANML	#CB
screen 4	#TC	#FF	#DWN	#ANML
screen 12	#SMII	#BLW	#CB	#TF
screen 5	#PH_CMBND	#BJD2	#BJD2	#DWN
screen 13	#TH	#PH_CMBND	#TF	#BJD2

The Movie Schedule by SilverScreeber Approach for Theatre 3 (Week 36-39)

Scr 11	382	PLANET OF THE APES	PLANET OF THE APES	THE FAST AND THE FURIOUS	MOULIN ROUGE
Scr 3	340	NYNKE	SWORDFISH	MOULIN ROUGE	THE FAST AND THE FURIOUS
Scr 1	223	DOWN TO EARTH	DOWN TO EARTH	RAT RACE	CAPTAIN CORRELLI'S MANDOLIN
Scr 2	223	DOWN	RAT RACE	SWORDFISH	RAT RACE
Scr 9	177	ANIMAL	THE CONTENDER	PLANET OF THE APES	SWORDFISH
Scr 10	177	BLOW	NYNKE	NYNKE	PLANET OF THE APES
Scr 8	173	FINAL FANTASY	DOWN	DOWN TO EARTH	DOWN TO EARTH
Scr 6	163	BRIDGET JONES DIARY	ANIMAL	THE CONTENDER	NYNKE
Scr 7	163	CRAZY/BEAUTIFUL	CRAZY/BEAUTIFUL	ANIMAL	THE CONTENDER
Scr 4	114	THE CONTENDER	FINAL FANTASY	DOWN	CRAZY/BEAUTIFUL
Scr 12	110	SCARY MOVIE II	BLOW	CRAZY/BEAUTIFUL	ANIMAL
Scr 5	104	TRAFFIC	BRIDGET JONES DIARY	BRIDGET JONES DIARY	TRAFFIC
Scr 13	100	PEARL HARBOUR	TRAFFIC	TRAFFIC	DOWN

APPENDIX-2 (A)

Lingo Program for Micro Scheduling

```
SETS:
TIME/1..50/;
SCREEN/1..6/:CAPS,VAR_COST;
MOVIE/1..6/:RUN;
MSCREEN(MOVIE,TIME):DEMAND;
MSTIME(MOVIE,SCREEN,TIME):START;
STIME(SCREEN,TIME);
ENDSETS
DATA:
RUN = 8 7 6 5 7 9;
CAPS = 80 100 110 140 150 200;
VAR_COST = 800 1000 1100 1400 1500 2000;
DEMAND = 990.96 990.96 990.96 990.96 990.96 990.96 990.96 990.96 990.96 990.96 1486.44
 1486.44 1486.44 1486.44 1486.44 1486.44 1486.44 1486.44 1486.44 1486.44 1734.18 1734.18
 1734.18 1734.18 1734.18 1734.18 1734.18 1734.18 1734.18 1734.18 743.22 743.22 743.22
 743.22 743.22 743.22 743.22 743.22 743.22 743.22 247.74 247.74 247.74 247.74 247.74 247.74
 247.74 247.74 247.74 247.74
 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20
 751.20 751.20 751.20 751.20 751.20 751.20 751.20 1126.80 1126.80 1126.80 1126.80 1126.80
 1126.80 1126.80 1126.80 1126.80 1126.80 563.40 563.40 563.40 563.40 563.40 563.40 563.40
 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40
 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20
 751.20 751.20 751.20 751.20 751.20 751.20 751.20 1126.80 1126.80 1126.80 1126.80 1126.80
 1126.80 1126.80 1126.80 1126.80 1126.80 563.40 563.40 563.40 563.40 563.40 563.40 563.40
 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40
 243.06 243.06 243.06 243.06 243.06 243.06 243.06 243.06 243.06 243.06 729.18 729.18 729.18
 729.18 729.18 729.18 729.18 729.18 729.18 729.18 972.24 972.24 972.24 972.24 972.24 972.24
 972.24 972.24 972.24 972.24 1701.42 1701.42 1701.42 1701.42 1701.42 1701.42 1701.42
 1701.42 1701.42 1701.42 1458.36 1458.36 1458.36 1458.36 1458.36 1458.36 1458.36 1458.36
 1458.36 1458.36
```

APPENDIX-2 (A)

Lingo Program for Micro Scheduling

```
SETS:
TIME/1..50/;
SCREEN/1..6/:CAPS,VAR_COST;
MOVIE/1..6/:RUN;
MSCREEN(MOVIE,TIME):DEMAND;
MSTIME(MOVIE,SCREEN,TIME):START;
STIME(SCREEN,TIME);
ENDSETS
DATA:
RUN = 8 7 6 5 7 9;
CAPS = 80 100 110 140 150 200;
VAR_COST = 800 1000 1100 1400 1500 2000;
DEMAND = 990.96 990.96 990.96 990.96 990.96 990.96 990.96 990.96 990.96 990.96 1486.44
1486.44 1486.44 1486.44 1486.44 1486.44 1486.44 1486.44 1486.44 1486.44 1734.18 1734.18
1734.18 1734.18 1734.18 1734.18 1734.18 1734.18 1734.18 1734.18 743.22 743.22 743.22
743.22 743.22 743.22 743.22 743.22 743.22 743.22 247.74 247.74 247.74 247.74 247.74 247.74
247.74 247.74 247.74 247.74
751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20
751.20 751.20 751.20 751.20 751.20 751.20 751.20 1126.80 1126.80 1126.80 1126.80 1126.80
1126.80 1126.80 1126.80 1126.80 1126.80 563.40 563.40 563.40 563.40 563.40 563.40 563.40
563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40
751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20 751.20
751.20 751.20 751.20 751.20 751.20 751.20 751.20 1126.80 1126.80 1126.80 1126.80 1126.80
1126.80 1126.80 1126.80 1126.80 1126.80 563.40 563.40 563.40 563.40 563.40 563.40 563.40
563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40 563.40
243.06 243.06 243.06 243.06 243.06 243.06 243.06 243.06 243.06 243.06 729.18 729.18 729.18
729.18 729.18 729.18 729.18 729.18 729.18 729.18 972.24 972.24 972.24 972.24 972.24 972.24
972.24 972.24 972.24 972.24 1701.42 1701.42 1701.42 1701.42 1701.42 1701.42 1701.42
1701.42 1701.42 1701.42 1458.36 1458.36 1458.36 1458.36 1458.36 1458.36 1458.36 1458.36
1458.36 1458.36
```

```
3805.83 3805.83 3805.83 3805.83 3805.83 3805.83 3805.83 3805.83 3805.83 3805.83 3262.14
3262.14 3262.14 3262.14 3262.14 3262.14 3262.14 3262.14 3262.14 3262.14 2174.76 2174.76
2174.76 2174.76 2174.76 2174.76 2174.76 2174.76 2174.76 2174.76 1631.07 1631.07 1631.07
1631.07 1631.07 1631.07 1631.07 1631.07 1631.07 1631.07 543.69 543.69 543.69 543.69 543.69
543.69 543.69 543.69 543.69 543.69
972.24 972.24 972.24 972.24 972.24 972.24 972.24 972.24 972.24 972.24 1458.36 1458.36
1458.36 1458.36 1458.36 1458.36 1458.36 1458.36 1458.36 1458.36 1701.42 1701.42 1701.42
1701.42 1701.42 1701.42 1701.42 1701.42 1701.42 1701.42 729.18 729.18 729.18 729.18 729.18
729.18 729.18 729.18 729.18 729.18 243.06 243.06 243.06 243.06 243.06 243.06 243.06 243.06
243.06 243.06;
ENDDATA

MAX = @SUM(MSTIME(I,J,T):(@SMIN(CAPS(J),DEMAND(I,T)/16))*16*START(I,J,T)-
VAR_COST(J)*START(I,J,T))-@SUM(TIME(T)| T # LE # 12 : 500*(1-
@SMIN(1,(@SUM(MSTIME(I,J,T1)|T1 # GE # 3*T # AND # T1 # LE # 3*T :
START(I,J,T1))))));
!CONSTRAINT FOR ENSURING THAT THE START VARIABLE IS BINARY
@FOR(MSTIME(I,J,T):@BIN(START(I,J,T)));
!THIS CONSTRAINT IS TOTAL NO OF MOVIE THAT CAN BE ASSIGNED AT ANY
TIME;
@FOR(TIME(T):@SUM(MSTIME(I,J,T):START(I,J,T))<=6);
!THIS CONSTRAINT IS FOR NOT SPLITTING THE MOVIE DURING ITS RUNLENGTH;
@FOR(STIME(J,T):@SUM(MSCREEN(I,T1)|T1 # GE # (T-RUN(I)+1) # AND # T1 # LE # T #
AND # T1 # GE # 1 : START(I,J,T1)) <= 1);
! THIS CONSTRAINT IS FOR ENSURING THAT NO TWO DIFFERENT MOVIE WILL
ASSIGN TO SAME SCREEN;
@FOR(MSCREEN(I,T):@SUM(STIME(J,T1)|T1 # GE # (T-RUN(I)+1) # AND # T1 # LE # T #
AND # T1 # GE # 1 : START(I,J,T1)) <= 1);
! THIS CONSTRAINT IS FOR STAFF;
@FOR(TIME(T):@SUM(MSTIME(I,J,T1)|T1 # GE # (T-RUN(I)) # AND # T1 # LE # (T-
RUN(I)) # AND # T1 # GE # 1 : START(I,J,T1)) <= 2);
!CONSTRAINT FOR ENSURING THAT EACH MOVIE WILL END BY THE CLOSE TIME
OF THE THEATRE;
@FOR(MSTIME(I,J,T1)|T1 # GE # (50-RUN(I)+2)# AND # T1 # LE # 50 : START(I,J,T1)=0);
```

APPENDIX-2 (B)

Code for Generating Various Scenario Problems for Micro-Scheduling

```
#include<stdlib.h>
#include<time.h>
#define TSLOT 50
int main()
{
    FILE *fp1,*fp2;
    int i,n,j,r,movie_type[15];
    float s1,s2,s3,s4,s5,pdemand[15],q1,q2,q3,q4,q5;
    float w_factor,w_demand[15];
    s1=0.7;s2=0.6;s3=0.4;s4=0.3;s5=0.1;
    fp2=fopen("microin8.txt","w+");
    fp1=fopen("weekdemand1.txt","r+");
    printf("enter the week-factor 0.3 for week days nad 0.7 for week end days\n");
    scanf("%f",&w_factor);
    printf("\n%f %f %f %f %f",s1,s2,s3,s4,s5);
    //printf("\nenter the no of movie");
    fscanf(fp1,"%d",&n);
    printf("\n the no of movies are %d",n);
    for(i=0;i<n;i++)
    fscanf(fp1,"%f",&w_demand[i]);
    for(i=0;i<TSLOT;i++)
    fprintf(fp2,"%-5d",i+1);
    fprintf(fp2,"\n");
    for(i=0;i<n;i++)
    {
        srand(0);
         r=rand()%5;
        movie_type[i]=r;
        fscanf(fp1,"%f",&w_demand[i]);
        printf("\n %d demand is %f",i, w_demand[i]);
        pdemand[i]=w_factor*w_demand[i];
        printf("\nThe movie type %d and demand is %f", movie_type[i],pdemand[i]);
        switch(movie_type[i])
        {
            case 0: {q1=s3;q2=s2;q3=s1;q4=s4;q5=s5; break;}
        case 1: {q1=s5;q2=s4;q3=s3;q4=s2;q5=s1; break;}
        case 2: {q1=s5;q2=s4;q3=s3;q4=s1;q5=s2; break;}
        case 3: {q1=s3;q2=s3;q3=s2;q4=s4;q5=s4; break;}
        case 4: {q1=s1;q2=s2;q3=s3;q4=s4;q5=s5; break;}
        }
        fprintf(fp2,"%d          ",i+1);
        printf("\n%f %f %f %f %f",q1,q2,q3,q4,q5);
        for(j=0;j<TSLOT;j++)
        {
                if(j<TSLOT/5)
           fprintf(fp2,"%-5.2f ",q1*pdemand[i]);
          else if(j<((2*TSLOT)/5))
```

```
                fprintf(fp2,"%-5.2f ",q2*pdemand[i]);
                else if(j<((3*TSLOT)/5))
                fprintf(fp2,"%-5.2f ",q3*pdemand[i]);
                else if(j<((4*TSLOT)/5))
                fprintf(fp2,"%-5.2f ",q4*pdemand[i]);
                else
                fprintf(fp2,"%-5.2f ",q5*pdemand[i]);
            }

            fprintf(fp2,"\n ");
        }
        return(0);
}
```

References

- Eliashberg, Jehoshua; Jonker, Jedid-Jah; Sawhney, Mohanbir S.; and Wierenga, Berend (2000), "MOVIEMOD: An implementable decision support system for pre-release market evaluation of motion pictures" *Marketing Science*, Volume 19, Number. 3, pp. 226-243.
- Eliashberg, Jehoshua and Mohanbir S. Sawhney (1994), "Modeling Goes to Hollywood: Predicting Individual Differences in Movie Enjoyment," *Management Science*, 40 (September),1151-1173.
- Eliashberg, Jehoshua; Swami, Sanjeev; Weinberg, Charles B. and Wierenga Berend (May-June2001), "Implementing and Evaluating SILVERSCREENER: A Marketing Management Support System for Movie Exhibitors," *Interfaces 31: 3.2(2)*, p.p. S108-S127.
- Eliashberg, Jehoshua and Steven M. Shugan (1997), "Film Critics: Influencers or Predictors?"*Journal of Marketing*, 61, 2 (April), 68-78.
- Weinberg, Charles B. (1986), "ARTS PLAN: Implementation, Evolution and Usage," *Marketing Science*, 5 (2), 143--158.
- Swami, Sanjeev, Eunkyu Lee, and Charles B. Weinberg (1998), "Optimal Channel Contracts for Marketing Perishable Products," Working Paper, University of British Columbia.
- Swami, Sanjeev, Jehoshua Eliashberg, and Charles B. Weinberg (1999), "SilverScreener: A Modeling Approach to Movie Screens Management," *Marketing Science*, 18 (3), pp. 352-372.
- Swami, Sanjeev (1998), "*Dynamic Marketing Decisions in the Presence of Perishable Demand*," Ph.D. Dissertation, University of British Columbia, Vancouver, Canada.
- *Variety, The International Entertainment Weekly.*
- Zufryden, Fred S. (1996), "Linking Advertising to Box Office Performance of New Film Releases-A Marketing Planning Model," *Journal of Advertising Research*, (July-August), 29-41.
- Goldberg, D.E. (1989), " Genetic Algorithms in Search, Optimization, and Machine Learning," *Addison-Wesley,Reading*, MA.

- Deb, Kalyanmoy (1996), " Optimization for Engineering Design: Algorithms and Examples," *Prentice Hall of India,* New Delhi.
- Ravindran, A., Philips, Don T. and Solberg, James J., (2000), " Operations Research: Principle and Practice," *John Wiley & Sons (ASIA) Pte Ltd*, Singapore.
- Acquilano, N. J., and R. B. Chase (1991), *Fundamentals of Operations Management*, Irwin, Homewood, IL.
- Bagchi, T.P. (1999), " Multiobjective Scheduling by Genetic Algorithms," Kluwer Academic Publishers, Dordrecht, The Netherlands.
- Bagchi, T.P., K. Deb, "Calibration of G.A. Parameters: The Design of Experiments Approach," *Computer Science & Informatics*, 26, 3.
- Michalewicz, Z.(1992), "Genetic Algorithms + Data Structures = Evolution Programs," *Springer-Verlag,* Berlin.
- Baker, Kenneth R. (1993), *Elements of Sequencing and Scheduling*, Dartmouth College, Hanover, NK.
- M Borin, Norm, Paul W. Farris, and James R. Freeland (1994), "A Model for Determining Retail Product Category Assortment and Shelf Space Allocation," *Decision Sciences*, 25 (3), 359--384.
- Bultez, Alain and Phillippe Naert (1988), "S.H.A.R.P.: Shelf Allocation for Retailers' Profit," *Marketing Science*, 7, 3 (Summer), 211--231.
- Moholkar, Makrand (2001), "Modelling and Heuristic Approaches to Retail Space Allocation in Complex Environments," Thesis Report, M.Tech., Indian Institute of Technology, Kanpur, India.
- Saxena, Sudhir (2000), "Implimentation of a Marketing Decision Support System for Motion Picture Retailing," Thesis Report, M.Tech., Indian Institute of Technology, Kanpur, India.
- Chauhan, Satyaveer Singh (2000), "Core Course Scheduling Using Genetic Algorithms ," Thesis Report, M.Tech., Indian Institute of Technology, Kanpur, India.
- Corstjens, M. and Doyle, P. (1981), " A Model for Optimizing Retail Space Allocations," *Management Science*, 27 (July), 822-833.
- Fourer, Robert, David M. Gay, and Brian W. Kernighan (1993), *AMPL: A Modeling Language for Mathematical Programming*, The Scientific Press, San Francisco, CA.

- Grefenstette, J. J.(1986), "Optimization of Control Parameters for Genetic Algorithms," *IEEE Trans. Systems, Man and Cybernetics*, SMC 16, 1, 122-128.
- Jain, Karuna and Edward A. Silver (1994), "Lot Sizing for a Product Subject to Obsolescence or Perishability," *European Journal of Operational Research*, 75 (2), 287--295.
- Jedidi, Kamel, Robert E. Krider, and Charles B. Weinberg (1998), "Clustering at the Movies," *Marketing Letters*, 9 (4), 393-405.
- Jones, Morgan and Christopher J. Ritz (1991), "Incorporating Distribution into New Product Diffusion Models," *International Journal of Research in Marketing*, 8, 91-112.
- Krider, Robert E. and Charles B. Weinberg (1998), "Competitive Dynamics and the Introduction of New Products: The Motion Picture Timing Game," *Journal of Marketing Research*, 35, 1 (February), 1-15.
- Kumar S., Bagchi, T.P., and Sriskandarajah (2000), "Lot streaming and scheduling heuristics for *m*-machine no-wait flowshops," *Computers and Industrial Engineering*, 38, 149-172.
- Lehmann, Donald R. and Charles B. Weinberg (2000), "Sales Via Sequential Distribution Channels: An Application to Movie Audiences," *Journal of Marketing*, 64 (3), 13-33.
- Lodish, Leonard M. (1971), "CALLPLAN: An Interactive Salesman's Call Planning System," *Management Science*, 18, 4(2), 25--40.
- Mahajan, Vijay, Eitan Muller, and Roger Kerin (1984), "Introduction Strategy for New Products with Positive and Negative Word-of-Mouth," *Management Science*, 30 (December), 1389-1404.
- Murata, T. and Ishibuchi,(1994) "Performance evaluation of genetic algorithms for flowshop scheduling problems," *Proceedings of the IEEE Conference on Genetic Algorithms*, 1994, 812-817.
- Prasad, Ashutosh, Vijay Mahajan, and Bart J. Bronnenberg (1998), "Product Entry Timing in Dual Distribution Channels: The Case of the Movie Industry," Working Paper, University of Texas at Austin.
- Radas, Sonja and Steven M. Shugan (1998), "Seasonal Marketing and Timing [illegible]" *Journal of Marketing Research*, 35 (August), 296-315.

A137960

- Sawhney, Mohanbir S. and Jehoshua Eliashberg (1996), "A Parsimonious Model for Forecasting Gross Box Office Revenues of Motion Pictures," *Marketing Science*, 15 (2), 113-131.